Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Vizualizacija medicinskih volumetričnih podatkov v realnem času

Žiga Lesar

Delo je pripravljeno v skladu s Pravilnikom o podeljevanju Prešernovih nagrad študentom, pod mentorstvom doc. dr. Matije Marolta.

Ljubljana 2014

Povzetek

V delu obravnavamo postopke za vizualizacijo medicinskih volumetričnih podatkov v realnem času, ki bi lahko povečali hitrost in zanesljivost diagnostičnih procesov. V ta namen smo razvili in primerjali tri vizualizacijske metode, ki temeljijo na algoritmu metanja žarkov: izris nivojskih ploskev z metodo regula falsi za povečanje natančnosti, projekcijo največje intenzitete in emisijsko-absorpcijski model za upodabljanje s prosojnostjo. Za izboljšanje dojemanja globine smo v upodobitev dodali tudi vizualna efekta ambientnega zastiranja in globinske ostrine. Vse metode smo optimizirali za čim hitrejše izvajanje na grafičnih procesorjih. Pri tem smo predstavili dve inovativni rešitvi: adaptivno vzorčenje in adaptivno rekonstrukcijo, ki pripomoreta k večji hitrosti izrisa in omogočata interaktivno vizualizacijo tudi na počasnejših napravah. Opravili smo performančno analizo vseh uporabljenih metod, ki pokaže na učinkovitost predstavljenih pohitritev ter na ozka grla pri implementaciji. Izvedli smo tudi uporabniško evalvacijo, v kateri so specialisti s področja radiologije ovrednotili primernost vizualizacijskih metod za iskanje podrobnosti in nepravilnosti na slikah ožilja. Evalvacija je potekala v obliki spletnega vprašalnika s primerjalnimi testi. Rezultati so pokazali, da se je najbolje odrezala metoda izrisa nivojskih ploskev z ambientnim zastiranjem, najslabše pa emisijsko-absorpcijski model.

Ključne besede: vizualizacija medicinskih podatkov, angiografija, metanje žarkov, nivojske ploskve, emisijsko-absorpcijski model, OpenCL.

Abstract

Our work discusses methods for real-time visualization of medical volumetric data sets, which can lead to faster and more reliable diagnostic processes. We have developed three visualization methods based on the ray casting algorithm: isosurface rendering with regula falsi intersection refinement, maximum intensity projection, and the emission-absorption model for rendering with transparency. For better depth perception we introduced two visual effects that enhance the final rendering: ambient occlusion and depth of field. We optimized the developed visualization methods for fast execution on graphics processing units and in the process introduced two novel methods: adaptive sampling and adaptive reconstruction, which contribute to faster rendering and allow for interactive visualization rates even on low-end devices. We conducted a performance evaluation of the methods, which showed the efficiency of the implemented speed-ups and revealed the bottlenecks in the implementation. We also carried out a user evaluation study, where radiology specialists evaluated the suitability of the visualization methods for presentation of details and detection of anomalies in angiographic images. The study has been conducted as an online survey with binary comparison tests and it shows that isosurface rendering with ambient occlusion is superior to other methods, especially the emission-absorption model.

Keywords: visualization of medical data, angiography, ray casting, isosurfaces, emission-absorption model, OpenCL.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
2D	two-dimensional	dvodimenzionalen
3D	three-dimensional	tridimenzionalen
BRDF	bidirectional reflectance distribution function	dvosmerna funkcija porazdelitve odbojnosti
BRE	balanced rank estimation	uravnoteženo ocenjevanje vrstnega reda
BSDF	bidirectional scattering distribution function	dvosmerna funkcija porazdelitve raztrosa
BSSRDF	bidirectional scattering-surface reflectance distribution function	dvosmerna funkcija porazdelitve raztrosa in odbojnosti
BTDF	Bidirectional transmittance distribution function	dvosmerna funkcija porazdelitve prepustnosti
CSG	constructive solid geometry	konstrukcija polnih geometrijskih teles
CT	computed tomography	računalniška tomografija
DOF	depth of field	globinska ostrina
FPS	frames per second	število slik na sekundo
GLSL	OpenGL shading language	jezik za pisanje OpenGL senčilnikov
HDRI	high-dynamic-range imagery	slike visokega dinamičnega razpona

HLSL	high-level shading language	visokonivojski jezik za pisanje senčilnikov
LMIP	local maximum intensity projection	projekcija lokalne največje intenzitete
MIP	maximum intensity projection	projekcija največje intenzitete
MLT	Metropolis light transport	transport svetlobe z algoritmom Metropolis
MPUI	multi-level partition of unity implicits	večnivojska delitev enote z uporabo implicitnih funkcij
MRI	magnetic resonance imaging	slikanje z magnetno resonanco
RGBA	red, green, blue, alpha	rdeča, zelena, modra, prosojnost
SSAO	screen-space ambient occlusion	slikovno-prostorsko ambientno zastiranje

Poglavje 1

Uvod

Vizualizacija tridimenzionalnih (3D) podatkov je postala nepogrešljivo orodje na področju znanosti, inženirstva, medicine, multimedijev in nasploh povsod, kjer predstavlja podlago za dojemanje prostora in vanj postavljenih predmetov. V znanosti pomaga pri vizualizaciji velikih količin podatkov in rezultatov najrazličnejših simulacij, ki pogosto temeljijo na fizikalnih pojavih, na primer toku tekočin, deformaciji predmetov in kemijskih reakcijah. V inženirstvu je temelj aplikacij za računalniško podprto načrtovanje, v multimedijih pa je 3D vizualizacija nepogrešljiva pri produkciji filmov, animacij in računalniških iger. Tudi na področju medicine se ljudje pri odločitvenih procesih in diagnosticiranju vse pogosteje opirajo na računalniško vizualizacijo, ki pripomore k neinvazivnem ali minimalno invazivnem razkritju notranjih struktur človeškega telesa.

V tehničnem smislu gre za sintezo dvodimenzionalnih (2D) slik s projiciranjem 3D predmetov iz prostora na projekcijsko ravnino. 3D predmeti so lahko predstavljeni na več načinov, najpogosteje kot poligonski modeli, lahko pa tudi implicitno ali parametrično. Za različne predstavitve predmetov obstajajo različni načini vizualizacije, prilagojeni omejitvam in prednostim določene predstavitve. Zaradi mnogih dobrih lastnosti trikotnika kot najpreprostejšega poligona se je sodobna 3D računalniška grafika usmerila v hitro izrisovanje velikega števila trikotnikov. Za obvladovanje tega pro-

blema so se v zadnjih desetletjih močno razvile grafične kartice - posebne računalniške komponente, namenjene prav hitremu izrisu trikotnikov na zaslon. V svojem bistvu so grafične kartice posebni podatkovno-pretokovni računalniki, ki so sposobni masovnega vzporednega računanja. Ta lastnost je pri mnogih računskih problemih izredno zaželena, saj lahko občutno zmanjša čas procesiranja. Prav zato je (v nasprotju s starejšo strojno opremo) na novejših grafičnih karticah mogoče izvajati tudi posebne programe - senčilnike, kar programerjem omogoča izkoriščanje zmožnosti vzporednega računanja za reševanje splošnejših problemov, ki niso omejeni zgolj na računalniško grafiko.

Vizualizacija v končni fazi pomeni prikaz podatkov v vizualni obliki - z barvo in intenziteto - na zaslonu. Človeški možgani pridobljene vizualne signale pretvarjajo v kompleksnejše oblike in koncepte, pri tem delujoči kognitivni procesi pa temeljijo na izkušnjah, spominu in pričakovanjih. Zato je smiselno, da računalniško generirane slike posnemajo lastnosti narave, v nekaterih primerih pa je zaželeno tudi, da vizualizacijo obogatimo s (sicer nerealističnimi) poudarki slikovnih značilk - oblik, robov ali celotnih predmetov.

V pričujočem delu se bomo posvetili predvsem vizualizaciji 3D volumnov. Volumni oz. volumetrične slike so diskretne predstavitve tridimenzionalnih podatkov, na primer rezultatov simulacij, meritev, medicinskih slik, pa tudi bolj abstraktnih podatkov, kot so denimo trirazsežne statistične porazdelitve. V nasprotju s poligonskimi modeli, ki služijo upodabljanju ploskev, so volumni namenjeni tako upodabljanju ploskev kot tudi notranjih struktur predmetov. Glavna prednost tega pristopa je hkratni prikaz različnih struktur istega predmeta na isti sliki, kar je še posebej uporabno za vizualizacijo medicinskih podatkov. Najpogostejše vrste medicinskih volumetričnih slik izvirajo iz naprav za računalniško tomografijo (angl. computed tomography - CT), magnetno resonanco (angl. magnetic resonance imaging - MRI) in 3D ultrazvok.

1.1 Cilji

Glavni cilj tega dela je razviti postopke za učinkovito interaktivno vizualizacijo medicinskih 3D volumnov, ki bodo optimizirani za vizualizacijo volumetričnih angiografskih slik. Pri angiografiji gre za minimalno invaziven postopek slikanja krvožilnega sistema, kjer v pacientov krvožilni sistem pred slikanjem vnesejo posebno barvilo. Tako pridobljena volumetrična slika vsebuje vrednosti, ki odražajo koncentracijo barvila v določeni točki. Postopek se pogosto uporablja pri diagnosticiranju raznih bolezni žil, kot so na primer anevrizme, uporaben pa je tudi pri pregledih pred raznimi operacijami.

Razviti postopki bi lahko poenostavili, izboljšali in povečali zanesljivosti postopkov diagnosticiranja anevrizem in drugih nepravilnosti krvožilnega sistema. Anevrizme so smrtno nevarne razširitve oslabljenih žilnih sten, ki pogosto nastanejo pri aterosklerozi. Nevarno je predvsem širjenje anevrizem, saj to lahko pripelje do močnih fizičnih pritiskov na sosednja tkiva, krvavitev in tromboze, ki so pogosti vzroki nenadne smrti. Hitra in zanesljiva diagnoza je zato ključnega pomena pri učinkovitem zdravljenju, k temu pa kot močna podlaga prispeva primerna vizualizacija.

Vizualizacija volumetričnih slik predstavlja ogromen tehničen izziv. Tovrstne slike dosegajo velikosti tudi več sto megabajtov, nekatere tudi več gigabajtov, zato je potrebno najprej razviti ustrezne metode, ki omogočajo učinkovito shranjevanje take količine podatkov, hkrati pa omogočajo hitre poizvedbe. V ta namen bomo razvili specializirane podatkovne strukture, ki z agregacijo podatkov nudijo uporabne informacije o območjih v volumnu. Prav tako kot volumetrični podatki bodo tudi te podatkovne strukture optimizirane za hitrost poizvedb, hkrati pa bodo potrebovale le malo dodatnega pomnilniškega prostora.

Velik izziv predstavlja tudi oblika informacij, ki jih dobimo iz naprav za volumetrično slikanje. Rezultat slikanja je namreč le skalarno polje, za interpretacijo podatkov pa je zadolžena aplikacija za vizualizacijo. V delu bomo zato razvili postopke za samodejno interpretacijo in klasifikacijo podatkov, hkrati pa bomo uporabniku dopustili možnost ročnega nastavljanja

parametrov teh postopkov, da bodo primerni za čim širšo uporabo.

Za uporabo v različne namene in poudarjanje različnih lastnosti volumnov imamo na voljo različne tehnike vizualizacije. Nekatere so bolj primerne za posredovanje topoloških informacij, druge za upodabljanje notranjih struktur. Obstajajo tudi metode, ki združujejo najboljše pristope, da lahko posredujejo več informacij o topologiji, globini, medsebojnih razdaljah, materialu in drugih lastnostih volumna. V delu bomo implementirali več pristopov in s primerjavo skušali objektivno prikazati razlike med tehnikami vizualizacije, dotaknili pa se bomo tudi vprašanj o praktični uporabni vrednosti pristopov.

V delu smo kot glavno tehniko vizualizacije izbrali metodo metanja žarkov in njene izpeljanke. Pri tej tehniki gre za iskanje presečišč med navideznimi žarki svetlobe in predmeti v prostoru. Za vsako zaslonsko točko moramo skozi kamero v prostor poslati vsaj en žarek, nato pa glede na njegovo interakcijo s predmeti generirati barvo, ki jo nato izrišemo na zaslon. Gre za sodobno metodo, s pomočjo katere lahko ustvarjamo izredno realistične slike, hkrati pa je to velik računski problem, saj za dovolj veliko sliko potrebujemo ogromno število žarkov. V diskretnih korakih na tem žarku računamo tudi vplive predmetov in materialov na svetlobo, kar še bistveno podaljša računski čas simulacije. Kljub temu lahko s pridom izkoriščamo medsebojno neodvisnost žarkov, zato se bomo v tem delu posvetili paralelizaciji in optimizaciji vseh opisanih metod za izvajanje na grafičnih karticah. Tako bomo bistveno pohitrili celoten proces vizualizacije, da bo primeren tudi za interaktivno uporabo. Metanje žarkov je mogoče izvajati v realnem času šele v zadnjih nekaj letih, zato se bomo srečevali s sodobnimi izzivi.

Ker želimo verodostojno simulacijo interakcije svetlobe z volumnom in posledično kvalitetno končno sliko, bomo opisali različne optične modele, ki temeljijo na obnašanju svetlobe v prosojnih materialih. Ker je za mnoge namene bolj primerno upodabljanje ploskev kot prosojnih materialov, bomo razvili tudi postopke za upodabljanje nivojskih ploskev. Za omenjene načine vizualizacije obstaja tudi vrsta pohitritvenih postopkov in struktur, ki jih bomo v želji po interaktivnosti vgradili v znane algoritme za metanje žarkov.

Del naloge bo posvečen tudi optimizaciji postopkov za angiografske volumetrične slike. Take slike namreč vsebujejo precej specifično vrsto podatkov, za katere bomo razvili posebne metode, ki bodo omogočale boljšo zaznavo značilk slike, oblik, globine in medsebojnih razdalj. Vse metode bodo prilagojene za odkrivanje anevrizem in podobnih nepravilnosti krvožilnega sistema.

Vse omenjene metode bomo združili v enotno aplikacijo, ki bo omogočala realno-časovno vizualizacijo velikih medicinskih slik za podporo diagnostičnim postopkom. Te postopke želimo poenostaviti in izboljšati, predvsem pa povečati njihovo zanesljivost. V ta namen bomo izvedli tudi evalvacijo, v kateri bodo strokovnjaki s področja radiologije ocenili uporabno vrednost razvitih metod.

1.2 Pregled področja

Vizualizacija volumetričnih podatkov ima že dolgo zgodovino. Osnovna metoda je metanje žarkov [4, 16], ki pa je le en od načinov vizualizacije volumetričnih podatkov. Algoritem generira barve zaslonskih točk s pošiljanjem navideznih žarkov svetlobe iz kamere v prostor in z računanjem interakcije teh svetlobnih žarkov s predmeti v prostoru. Algoritem spada med t.i. neposredne metode, saj podatkov pred vizualizacijo ne pretvorimo v drugačno obliko. Med neposrednimi metodami so znane in v široki uporabi še vizualizacija s tridimenzionalnimi teksturami [43], algoritem shear-warp [13] in splatting [47]. Vsak pristop je v svoji zasnovi popolnoma drugačen, toda vsem je skupno vzorčenje volumetričnih podatkov in združevanje barv na podoben način kot pri metanju žarkov.

Najpreprostejši pristop, ki je podprt tudi na manj zmogljivih grafičnih karticah, je upodabljanje volumnov s tridimenzionalnimi teksturami. Gre za upodabljanje množice v kamero usmerjenih rezin, ki jih senčimo z vzorčenjem barv iz tridimenzionalne teksture. Pristop je sicer hiter, toda ima več pomanjkljivosti, predvsem natančnost izrisa in nezmožnost uporabe učinkovitih podatkovnih struktur in naprednejših senčilnih tehnik. Primerljivo hiter,

toda kvalitetnejši pristop, je algoritem shear-warp. Celoten volumen se pred projekcijo (v fazi *shear*) transformira tako, da so rezine volumna poravnane z dvema koordinatnima osema. Po taki transformaciji sta vzorčenje in sestava barv precej hitrejša, je pa zaradi tega slika popačena. V drugi fazi (*warp*) se popačena slika transformira v končno, nepopačeno obliko. Še hitrejša metoda je splatting, pri kateri na račun hitrosti trpi kvaliteta slike. Pri tej metodi se vzorci volumna transformirajo s pogledom, uredijo glede na oddaljenost od kamere, nato pa se na zaslon izriše barvna pika Gaussove oblike in primerne velikosti. Metoda je sicer bolj primerna za upodabljanje sistemov delcev, uporabna pa je tudi za vizualizacijo volumetričnih podatkov.

V nasprotju z neposrednimi metodami posredne metode volumetrične podatke pretvorijo v drugačno predstavitev, najpogosteje v poligonski model. Dve najpogostejši metodi za t.i. poligonizacijo volumnov sta marching cubes [21] in Bloomenthalov poligonizator [2]. Slednji se uporablja precej redkeje, saj za algoritem marching cubes obstajajo izredno hitre implementacije. Algoritem marching cubes je za vzporedno izvajanje na grafični kartici v svoji diplomski nalogi predstavil Anže Sodja [41]. Težava omenjenih algoritmov je v nezmožnosti modeliranja ostrih značilnosti predmetov. Leta 2003 je to težavo deloma rešil algoritem MPUI (angl. multi-level partition of unity implicits), ki uporablja implicitne funkcije za modeliranje lokalne ukrivljenosti predmeta [29]. Vhod v algoritem je oblak točk v prostoru, torej moramo volumen v tako obliko pretvoriti še pred uporabo MPUI algoritma. Izhod algoritma je skalarno polje, ki hrani predznačene razdalje do površine predmeta. Ta predstavitev je primerna za sorodno metodo metanju žarkov, sphere tracing [10], ki tako skalarno polje izkorišča za učinkovito preskakovanje praznega prostora.

Naštete posredne metode so zaradi arhitekture sodobnih grafičnih kartic izredno primerne za izris, toda faza pretvorbe, ki pri neposrednih metodah ni prisotna, porabi več časa. Poleg tega je izhod takih metod pogosto poligonski model, kar je sicer odlično za upodabljanje ploskev, toda precej manj primerno za upodabljanje prosojnih materialov. Prav zato je standardna

izbira za vizualizacijo medicinskih podatkov metanje žarkov, saj tako lažje razkrijemo notranjo strukturo predmetov.

Začetki algoritmov metanja žarkov za upodabljanje volumetričnih podatkov segajo v osemdeseta leta dvajsetega stoletja. Leta 1982 je Scott Roth predstavil osnovni algoritem metanja žarkov [37], ko je z njegovo uporabo upodabljal modele, zgrajene z metodo CSG (angl. constructive solid geometry). Njegovo metodo je Marc Levoy leta 1988 razširil za uporabo na volumetričnih podatkih [16]. Kasneje je svojo metodo prilagodil za hkratno upodabljanje volumnov in poligonskih modelov [17]. Implementiral je tako izris nivojskih ploskev kot tudi prosojnih materialov z uporabo Porterjevega operatorja za sestavljanje barv [34]. Omenjeni operator je rezultat diskretizacije zveznega optičnega modela, podrobneje predstavljenega v [24], kjer so opisani osnovni optični modeli za upodabljanje volumnov. John Hart je leta 1993 objavil članek, v katerem je predstavil aplikacijo intervalske analize in Lipschitzove lastnosti za iskanje nivojskih ploskev [9]. Za povečanje natančnosti najdenih presečišč je uporabil Newtonovo metodo in metodo regula falsi. Slednjo smo uporabili tudi v pričujočem delu.

Algoritem za metanje žarkov potrebuje ogromno vzorcev volumna, zato so se že zgodaj pojavile številne metode in podatkovne strukture za pospešitev procesa. Med najbolj razširjenimi je še dandanes osmiško drevo (angl. octree) in njegove različice. Osmiško drevo, predstavljeno v [25, 6], spada v kategorijo metod za razdelitev prostora. Pri upodabljanju nivojskih ploskev služi učinkovitemu preskakovanju praznega prostora, pri upodabljanju prosojnih materialov pa hitrejšemu procesiranju homogenih območij. Med drugimi razširjenimi pristopi so tudi k-d drevesa in binarno razdeljevanje prostora, ki pa so v primerjavi z osmiškim drevesom precej redkeje uporabljeni v praksi.

Osmiška in k-d drevesa so skozi leta doživela mnogo izboljšav in prilagoditev za vizualizacijo volumnov. Na tem področju izstopajo predvsem dela Tima Foleya, Daniela Madeire in Samulija Lainea. Njihova spoznanja temeljijo na dejstvu, da so grafične kartice neprimerne za sekvenčna opravila, ki pa so pri sprehajanju in iskanju po drevesih nujna. Foley v članku [5] predstavi

dve hitri implementaciji za sprehod po k-d drevesu, primerni za računanje na grafičnih karticah. Madeira v članku [23] opiše nekaj metod za optimizirano iskanje po osmiškem drevesu, hkrati pa predstavi nove načine kodiranja osmiških dreves, ki omogočajo vzporedno iskanje večje množice točk. Leta 2011 je Samuli Laine v članku [14] predstavil metode za učinkovito kodiranje topoloških informacij, barv in normal v osmiškem drevesu. Za sprehod po drevesu je v članku uporabljen algoritem, prevzet po [35]. Podobna različica algoritma je uporabljena tudi v pričujočem delu. Laineov članek postavlja nove mejnike za upodabljanje volumnov v realnem času, toda za potrebe pričujočega dela je bil neprimeren, saj ni zmožen upodabljanja prosojnih materialov. Dober pregled upodabljanja volumnov na grafični kartici in algoritmov za sprehod po osmiškem drevesu poda tudi Kristof Römisch v svoji magistrski nalogi [36]. Njegove ugotovitve glede hitrosti algoritmov smo upoštevali pri implementaciji osmiških dreves.

Glavne izboljšave hitrosti upodabljanja so plod boljših algoritmov in podatkovnih struktur, obstajajo pa tudi drugačni pristopi. V članku iz leta 1997 [48] sta opisani dve metodi za izkoriščanje časovne koherence pri metanju žarkov: slikovni predpomnilnik in nivojski zemljevid. Metodi zmanjšata število vzorčenj preden žarek preseka nivojsko ploskev, hkrati pa izkoriščata dokaj zvezno spreminjanje slike pri kratkih premikih kamere. Koherenco končne slike izkorišča tudi metoda, opisana v [12]. Gre za adaptivno metanje žarkov na podlagi ocene koherence lokalnih slikovnih značilnosti. Metoda je odlično skalabilna z večanjem ločljivosti slike, kar je pri sodobnih zaslonih visoke ločljivosti dobrodošlo. Kot izboljšava algoritma Marca Levoya [18] je metoda primerna tudi za izris prosojnih materialov. Poenostavljeno različico metode smo implementirali tudi za potrebe tega dela.

Optimizacijam hitrosti zaradi vse večjih volumetričnih slik sledijo izboljšave porabe pomnilnika. Grimm v članku iz leta 2004 [8] opiše metode za učinkovito vizualizacijo volumnov na centralni procesni enoti, hkrati pa so predstavljene metode tudi pomnilniško učinkovite, saj izrabljajo predpomnilniške podatkovne strukture. Ker se pri vzorčenju upoštevajo le lokalni

vzorci signala, je shranjevanje celotnega volumna v pomnilniku lahko potratno in neskalabilno. V [7] so predstavljene učinkovite podatkovne strukture in sistemi za ločeno upravljanje s pomnilnikom, ki dele volumna po potrebi prenesejo s trdega diska na grafično kartico. Omenjene pristope s tehnikami iz [14] združi Felix Weißig v svoji diplomski nalogi, kjer je predstavljena tudi podrobna in obširna analiza rezultatov [46].

Veliko dela je bilo opravljenega na račun boljše in pravilnejše osvetljave predmetov. Človeško oko je namreč prilagojeno sklepanju o globinskih in topoloških informacijah na podlagi senc in nihanj intenzitete svetlobe, zato so bile razvite tudi temu primerne tehnike za osvetlitev in senčenje. Volumni s prosojnostjo prinašajo nove izzive, povezane z optičnimi modeli in učinkovitostjo njihovega računanja. V podjetju Pixar so razvili metodo za volumetrično vizualizacijo senc, imenovano deep shadow maps [20]. Primerna je za vrsto različnih pojavov, med drugimi tudi za volumetrične podatke, sisteme delcev in zabrisano gibanje. Poleg senc so izboljšave nastopile tudi pri metodah senčenja. Starejše in dobro poznane metode so prilagojene za izris neprosojnih ploskev, zato ne pridejo v poštev pri volumnih, saj se svetloba namreč v takih materialih obnaša povsem drugače. Andrea Kratz v svoji diplomski nalogi [11] opiše različne tehnike osvetljevanja in senčenja za volumne, med drugim tudi prilagodi metodo deep shadow maps za izvajanje na grafični kartici. Predstavljene metode so nadvse uporabne pri senčenju homogenih prosojnih območij, saj ustaljeni pristopi povzročajo neželena popačenja zaradi nezanesljive ocene gradienta.

Najnovejši pristopi za osvetljevanje izkoriščajo statistične metode za konvergenco h končnemu rezultatu. Med njimi izstopa sledenje svetlobi (angl. path tracing), ki v nasprotju s sledenjem žarkom (angl. ray tracing) svetlobne žarke pošilja iz virov svetlobe in ne iz kamere. Metoda dandanes velja za najboljšo in se kot standardna izbira široko uporablja v industriji. Sorodna metoda, zemljevid fotonov (angl. photon mapping), je bila nedavno prilagojena tudi za uporabo na volumetričnih podatkih [49]. Statičen zemljevid fotonov se zgradi pred vizualizacijo, zato pristop ni primeren za dinamično

osvetljevanje.

Pogosta metoda za aproksimacijo globalne osvetlitve je tudi ambientno zastiranje oz. aproksimacija dostopnosti površine (angl. ambient occlusion, surface accessibility). Korenine te široko uporabljane metode segajo v leto 1994, ko je Gavin Miller v članku [26] predstavil algoritem za računanje lokalne in globalne dostopnosti. Pristopi so zaradi svoje računske zahtevnosti le delno uporabni za interaktivne aplikacije, zato obstaja grob približek, ki se računa na končni sliki s pomočjo globinske slike - tehnika je slikovno-prostorska. Razvili so jo v podjetju Crytek leta 2007 in jo poimenovali SSAO (angl. screen-space ambient occlusion). V primerjavi s pravim ambientnim zastiranjem ima slikovno-prostorska tehnika močne prednosti, med drugimi hitrost in neodvisnost od kompleksnosti prizorišča. Različico metode smo implementirali tudi v tem delu. Obstajajo tudi izboljšave, ki upoštevajo lokalno normalo [39, 44] - slika normal je običajno na voljo pri aplikacijah, ki uporabljajo odloženo upodabljanje (angl. deferred shading). Primer implementacije in dober pregled metod najdemo v diplomskem delu Marka Tatića [42].

1.3 Struktura dela

Delo je strukturirano v pet poglavij. Uvodu sledi poglavje 2, kjer so predstavljeni koncepti, algoritmi, pohitritvene strukture in vizualni efekti, razviti za vizualizacijo podatkov. Sledita poglavji o implementaciji in evalvaciji predstavljenih rešitev, v zaključku pa povzamemo rezultate dela in njegove originalne prispevke ter podamo ideje za nadaljnji razvoj.

Poglavje 2

Metode in orodja

V poglavju bomo predstavili algoritmično ozadje vizualizacije 3D volumnov. Sprva bomo podali nekaj osnov tridimenzionalne računalniške grafike, ki so temelj za uporabljeni algoritem metanja žarkov. Uvedli bomo koncept kamere in s tem postopek projiciranja 3D predmetov na projekcijsko ravnino kamere (zaslon). Poglobili se bomo v strukturo volumetričnih podatkov v pomnilniku in kako nam algoritem metanja žarkov pomaga pri rekonstrukciji 3D struktur iz teh podatkov. Rekonstrukcijo bi radi implementirali tako, da bo posredovala čim več informacij o oblikah, globini, prekrivanju struktur in medsebojnih razdaljah. Ker ljudje veliko tovrstnih informacij dobijo prek osvetlitve predmetov v prostoru, bomo opisali našo rešitev za osvetljevanje.

V postopku klasifikacije bomo območjem v volumnu določili tudi optične lastnosti, ki služijo bolj verodostojni simulaciji svetlobe. Ker bi radi dosegli čim bolj realistično simulacijo interakcije svetlobe z materiali, bomo predstavili nekaj optičnih modelov. Ti modeli so namenjeni vizualizaciji prosojnih volumetričnih materialov, kar je zelo zaželeno pri diagnostičnih postopkih, saj pogosto pripomore k hitrejši in zanesljivejši diagnozi.

Algoritem metanja žarkov za kvaliteten izris slike potrebuje izračun velike količine svetlobnih žarkov, ki skupaj s kompleksnimi optičnimi modeli vodijo do velike računske komplekstnosti algoritma, zato si bomo kasneje pogledali tudi razne pohitritvene strukture in algoritme (angl. accelerating structures

and algorithms). Radi bi namreč dosegli interaktivnost aplikacije, saj animacija pripomore k boljši prostorski predstavi. Ker so v medicinskih slikah in predvsem v angiografskih slikah prisotna velika prazna območja, bomo uporabili drevesno strukturo, ki nam bo omogočala učinkovito preskakovanje praznih in homogenih območij. V območjih, kjer bo vzorčenje volumna še vseeno nujno potrebno, bomo uporabili adaptivno rekonstrukcijo, ki bo pripomogla k bistveno hitrejši rekonstrukciji volumetričnih struktur z minimalnim vplivom na kvaliteto končne slike. Pri prosojnih materialih, kjer je gosto vzorčenje pogosto ključno za kvalitetno upodobitev, bomo uporabili pristop z adaptivnim vzorčenjem. Tako bomo tista območja volumna, ki so za uporabnika bolj pomembna, upodabljali bolj natančno. S tem bodo izgube na kvaliteti minimalne, ogromno pa bomo pridobili na interaktivnosti. Za še hitrejšo vizualizacijo bomo neposredno zmanjšali število žarkov z metodo redkega metanja žarkov. Tako bomo žarek poslali za vsako drugo zaslonsko točko, v vmesnih prostorih pa bomo barve interpolirali.

Nekaj govora bo tudi o vizualnih efektih. Gre za enostavne efekte s področja obdelave slik, ki pa lahko pozitivno vplivajo na uporabnikovo percepcijo, zato smo se odločili, da dva efekta preizkusimo tudi v naši aplikaciji. Za poudarjanje oblik in globine smo uporabili pristop z ambientnim zastiranjem, ki upodablja sence tako, da poudarja vpliv bližine predmetov na svetlost površin. Za poudarjanje globine in medsebojnih razdalj ter za usmerjanje uporabnikove pozornosti smo se odločili za uporabo efekta globinske ostrine. Efekta sta zelo pogosta v animaciji in računalniških igrah, kjer kljub svoji enostavnosti bistveno pripomoreta h kvaliteti končne slike. Dokaj pogosto ju zasledimo tudi v filmih, saj posredujeta veliko informacij o predmetih v prostoru.

2.1 Tridimenzionalna računalniška grafika in postopek metanja žarkov

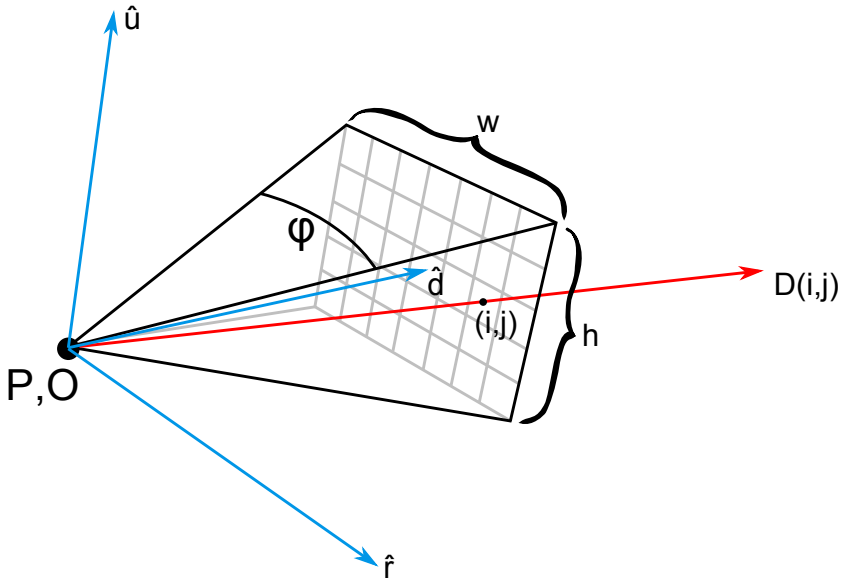
Tridimenzionalna računalniška grafika se ukvarja s predstavitvijo, manipulacijo in prikazovanjem tridimenzionalnih predmetov na dvodimenzionalni projekcijski ravnini (zaslonu). Sodoben pristop temelji na množici točk, ki jih z nelinearno projekcijsko transformacijo preslikamo na zaslon.

Ključen element prikazovanja 3D predmetov je navidezna kamera. Kamera je predstavljena s četvorčkom $(\mathbf{P}, O, \varphi, \alpha)$, kjer je $\mathbf{P} \in \mathbb{R}^3$ položaj v prostoru, O orientacija, φ zorni kot in α projekcijsko razmerje med horizontalo in vertikalo projekcije. Orientacija v prostoru je pogosto izražena s kvaternionom ali Eulerjevimi koti. V tem delu smo se odločili za enostavnejšo toda bolj omejeno rešitev, saj je implementacija enostavnejša, interakcija s kamero pa bolj intuitivna - orientacijo kamere smo predstavili z dvema sferičnima kotoma. Ne glede na izbrano predstavitev orientacije lahko določimo novo bazo prostora. Kamera v svojem lokalnem prostoru gleda v smeri $+z$ v levosučnem koordinatnem sistemu, zasukana pa je tako, da smer $+y$ pomeni smer navpično navzgor. Novo ortonormirano bazo prostora tako določajo vektorji ($\hat{\mathbf{r}}$, $\hat{\mathbf{u}}$, $\hat{\mathbf{d}}$), ki v lokalnem prostoru kamere predstavljajo standardno bazo ($\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, $\hat{\mathbf{k}}$). Kamera in njeni atributi so prikazani na sliki 2.1.

Metanje žarkov je v kontekstu računalniške grafike postopek iskanja projekcije tridimenzionalnih predmetov na dvodimenzionalno projekcijsko ravnino z iskanjem presečišč žarkov svetlobe s predmeti na prizorišču. Bistvo metode je iskanje kameri najbližjega presečišča, s čimer rešimo vprašanje prekrivanja projekcij. Metoda sestoji iz dveh glavnih delov: konstrukcija žarkov in iskanje presečišč. Žarek je enodimenzionalen matematični konstrukt, ki ga lahko predstavimo s funkcijo $\mathbf{r} : \mathbb{R} \to \mathbb{R}^3$, ki parametru priredi položaj v prostoru. Najbolj enostavna parametrizacija je

$$\mathbf{r}(\lambda) = \mathbf{P} + \lambda \hat{\mathbf{D}}(i, j), \tag{2.1}$$

kjer je $\mathbf{r}$ točka na žarku, $\lambda \geq 0$ prosti parameter in $\hat{\mathbf{D}}(i, j)$ normaliziran



Slika 2.1: Shematična predstavitev kamere in njenih atributov.

smerni vektor žarka, ki je odvisen od položaja (i, j) zaslonske točke.

V fazi konstrukcije za vsako zaslonsko točko (i, j) zaslona dimenzij (w, h) izračunamo smerni vektor iz ortonormirane baze po naslednji formuli:

$$\mathbf{D}(i,j) = \hat{\mathbf{d}} + (\frac{i}{w} - \frac{1}{2}) \tan\frac{\varphi}{2}\hat{\mathbf{r}} + (\frac{j}{h} - \frac{1}{2})\alpha \tan\frac{\varphi}{2}\hat{\mathbf{u}} \tag{2.2}$$

$$\hat{\mathbf{D}}(i,j) = \frac{\mathbf{D}(i,j)}{||\mathbf{D}(i,j)||} \tag{2.3}$$

Tako konstruiran žarek nato uporabimo v fazi iskanja presečišč, kjer iščemo najmanjšo vrednost parametra λ, ki odraža točko na iskani površini. Pogoj, da je vrednost parametra najmanjša možna, rešuje problem prekrivanja projekcij. Metode iskanja presečišč z volumetričnimi podatki so opisane v poglavju 2.4. Ko najdemo točko presečišča, lahko s tehnikami senčenja pridobimo njeno barvo, ki jo prikažemo na zaslonu v točki (i, j).

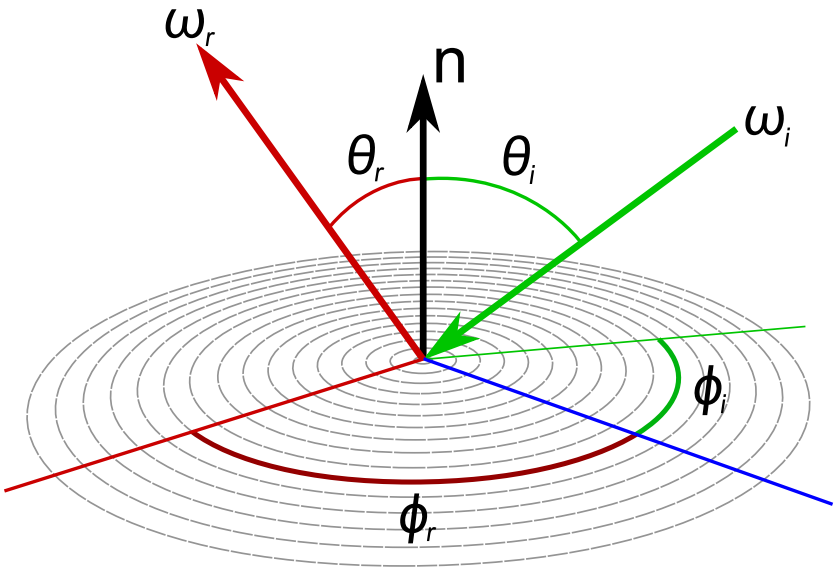
2.2 Senčenje

Senčenje je v tridimenzionalni računalniški grafiki postopek pridobivanja barvne informacije na podlagi lastnosti površine, kamere in luči v prostoru z namenom posredovanja globinske in topološke informacije. Obstaja veliko različnih modelov senčenja, toda v interaktivnih aplikacijah se najpogosteje uporabljajo Lambertov, Phongov in Blinn-Phongov model. Phongov model [33] je le razširitev Lambertovega modela z dodanim členom za modeliranje zrcalno odbite svetlobe, Blinn-Phongov model [1] pa je poenostavitev Phongovega modela, kjer je člen zrcalno odbite svetlobe izračunan nekoliko drugače. Obstajajo tudi kompleksnejši modeli, na primer Cook-Torranceov model [3] in Oren-Nayarjev model [30], toda za potrebe interaktivne računalniške grafike so zaradi daljših izračunov neprimerni.

Pogosto se za natančnejše modeliranje materialov uporabljajo funkcije BRDF (angl. bidirectional reflectance distribution function), predstavljene v [28]. Definirane so kot

$$f_r(\omega_i, \omega_r) = \frac{\mathrm{d}L_r(\omega_r)}{E_i(\omega_i)} = \frac{\mathrm{d}L_r(\omega_r)}{L_i(\omega_i)\cos\theta_i \ \mathrm{d}\omega_i}, \tag{2.4}$$

kjer je ω_i vpadna smer, ω_r odbojna smer, L_r intenziteta svetlobe v odbojni smeri, L_i intenziteta svetlobe v vpadni smeri, E_i osvetljenost v vpadni smeri in θ_i vpadni zenitni kot. Funkcija $f_r(\omega_i, \omega_r)$ opisuje odboj svetlobe glede na vpadno in odbojno smer. Ker je vsaka smer opisana z azimutom ϕ in zenitnim kotom θ, je funkcija štiridimenzionalna. Parametri funkcije BRDF so prikazani na sliki 2.2. Računanje s funkcijo BRDF je enostavnejše, saj ne potrebujemo težjih vektorskih operacij. Uporabimo lahko namreč preslikovalno tabelo. Težava nastopi pri frekvenci vzorčenja in dimenzionalnosti funkcije, saj za zadovoljive rezultate potrebujemo dokaj veliko tabelo. To je bil tudi glavni razlog, da smo uporabo funkcije BRDF opustili. Obstajajo tudi posplošitve, ki poleg odbojev svetlobe modelirajo tudi svetlobne pojave pod površino. To so funkcije BSDF (angl. bidirectional scattering distribution function), BSSRDF (angl. bidirectional scattering-surface reflectance



Slika 2.2: Vizualna predstavitev parametrov funkcije BRDF.

distribution function) in BTDF (angl. bidirectional transmittance distribution function), toda zaradi večje dimenzionalnosti so dandanes te funkcije praktično neprimerne za interaktivno uporabo.

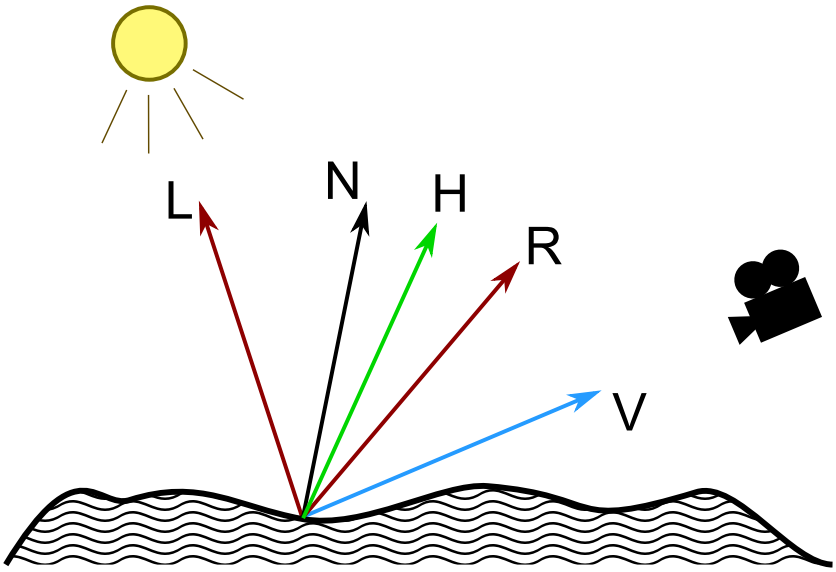
Phongov model, ki smo ga v nekoliko poenostavljeni različici uporabili tudi v pričujočem delu, izračuna barvo $\mathbf{c}$ točke kot

$$\mathbf{c} = \sum_{m \in lights} \left[\mathbf{k}_A \mathbf{i}_{m,A} + \mathbf{k}_D \mathbf{i}_{m,D} (\hat{L}_m \cdot \hat{N}) + \mathbf{k}_S \mathbf{i}_{m,S} (\hat{R}_m \cdot \hat{V})^\sigma \right]. \tag{2.5}$$

V formuli veljajo naslednji paramteri materiala:

- k_A je ambientna svetloba,
- k_D je barva razpršeno odbite svetlobe,
- k_S je barva zrcalno odbite svetlobe,
- σ je gladkost materiala.

Veljajo tudi naslednji parametri luči m:



Slika 2.3: Vizualna predstavitev vektorjev, prisotnih pri izračunu Phongovega modela osvetljevanja.

- $i_{m,A}$ je ambientna barva luči,
- $i_{m,D}$ je barva luči, ki se uporablja pri razpršenem odboju,
- $i_{m,S}$ je barva luči, ki se uporablja pri zrcalnem odboju.

Poleg parametrov materiala in luči se pojavljajo še druge vrednosti:

- $\hat{N}$ je enotski vektor normale na ploskev v presečišču,
- $\hat{V}$ je vektor pogleda - enotski vektor, ki je usmerjen od presečišča do kamere,
- $\hat{L}_m$ je vektor luči - enotski vektor, ki je usmerjen od presečišča do luči m,
- $\hat{R}_m$ je vektor luči, prezrcaljen preko normale $\hat{N}$.

V Phongovem členu se $\hat{R}_m$ izračuna kot

$$\hat{R}_m = 2(\hat{L}_m \cdot \hat{N})\hat{N} - \hat{L}_m. \tag{2.6}$$

V vsoti je prvi člen ambientni člen, ki posnema ambientno osvetlitev. Ta člen se vsoti doda s faktorjem 1, saj je neodvisen od vseh ostalih parametrov prizorišča. Drugi člen je Lambertov člen, ki modelira intenziteto razpršeno odbite svetlobe idealnega grobega materiala (na primer papir, les) po Lambertovem kosinusnem zakonu[1]. Ta pravi, da je intenziteta odbite svetlobe sorazmerna s kosinusom vpadnega kota. Vektor pogleda v tem členu ne nastopa, kar posledično pomeni, da taki materiali izgledajo enako ne glede na položaj kamere. Zadnji člen je Phongov člen, ki modelira močnejšo intenziteto svetlobe blizu odbojne smeri. Primeren je za materiale kot so polirana kovina, plastika in lakirani materiali. V Blinn-Phongovem modelu se namesto $\hat{R}_m \cdot \hat{V}$ uporablja $\hat{H}_m \cdot \hat{N}$, kjer se kompromisni vektor $\hat{H}_m$ izračuna kot

$$\hat{H}_m = \frac{\hat{L}_m + \hat{V}}{||\hat{L}_m + \hat{V}||}. \tag{2.7}$$

Vsi vektorji, ki so prisotni v izračunih barve s Phongovim modelom, so prikazani na sliki 2.3.

Pogosto se modelira tudi upadanje intenzitete svetlobe tako, da se prispevke luči pomnoži še s faktorjem $(a_0 + a_1 d + a_2 d^2)^{-1}$, kjer je d razdalja točke do luči, a_i pa koeficienti, ki natančneje določajo moč upadanja intenzitete. V tem delu smo ta faktor izpustili, saj nima praktične vrednosti pri upodabljanju angiogramov.

2.3 Vzorčenje in rekonstrukcija signala

Ker gre pri volumetričnih podatkih za diskreten zajem vrednosti pri ustvarjanju 3D slike ter kasneje pri rekonstrukciji in projekciji slike na zaslon, se moramo zavedati predpostavk in omejitev, ki jih taki postopki prinašajo.

[1] `http://en.wikipedia.org/wiki/Lambert's_cosine_law`

Ko govorimo o interaktivni grafiki, je poznavanje matematičnega ozadja postopkov vzorčenja še posebej pomembno, saj je v tem primeru natančna rekonstrukcija časovno zelo potratna.

V nadaljevanju bomo z izrazom *izvorni signal* označili realen signal, ki ga poskušamo diskretizirati s postopki medicinskega slikanja. Vzorčenje izvornega signala je že del naprav za zajemanje slike (najpogosteje CT in MRI) in nanj pri vizualizaciji nimamo vpliva. Imamo pa velik vpliv tako na postopek rekonstrukcije izvornega signala iz diskretiziranih podatkov kot tudi na postopek ponovnega vzorčenja tega signala v diskretnih točkah na svetlobnem žarku. Izvorni signal je v diskretnih točkah v prostoru tudi kvantiziran, običajno z 8-bitno ali 16-bitno natančnostjo. Vse to prispeva k manj natančni rekonstrukciji, kar pa je za hitro vizualizacijo žal nujno zlo.

Volumen je v matematičnem smislu diskretno vzorčenje funkcije oz. izvornega signala $\Phi : \mathbb{R}^3 \to \mathbb{R}$, ki vsako točko v prostoru preslika v realno vrednost. Ta vrednost je povsem abstraktnega pomena, odvisno od aplikacije, pogosto pa odraža gostoto ali koncentracijo. Kot za vsako vzorčenje tudi v tem primeru velja, da je popolna rekonstrukcija izvornega signala možna le, če je frekvenca vzorčenja vsaj dvakrat tolikšna kakor najvišja frekvenca, prisotna v izvornem signalu. V eni dimenziji je ta lastnost vzorčenja znana kot Shannon-Nyquistov izrek [40], v višjih dimenzijah pa velja posplošitev, znana kot Petersen-Middletonov izrek [32]. Z uporabo Whittaker–Shannonove interpolacijske formule [40] lahko izvorni enodimenzionalni signal natančno rekonstruiramo s konvolucijo z inverzno Fourierovo transformacijo nizkoprepustnega filtra

$$H(\nu) = \operatorname{rect}\left(\frac{\nu}{2W}\right), \tag{2.8}$$

$$\begin{aligned} h(t) &= \mathcal{F}^{-1}\{H(\nu)\} && (2.9)\\ &= 2W\frac{\sin(2\pi Wt)}{2\pi Wt} && (2.10)\\ &= 2W\operatorname{sinc}(2Wt), && (2.11) \end{aligned}$$

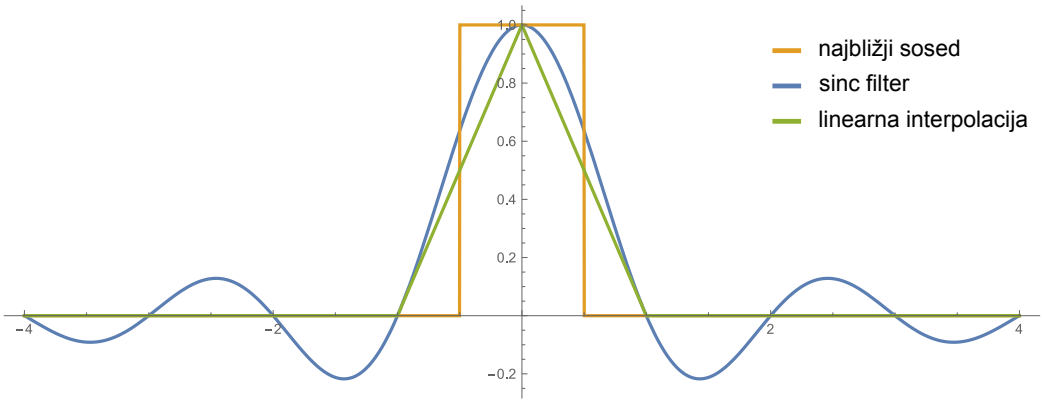
kjer je t čas, ν frekvenca, $h(t)$ rekonstrukcijski filter, $H(\nu)$ nizkoprepustni filter in W najvišja frekvenca, prisotna v izvornem signalu [31]. Za posplošitev v višje dimenzije lahko uporabimo tenzorski produkt. Pri taki rekonstrukciji pride do naslednjih težav:

- Funkcija $h(t)$ je definirana na celotni množici realnih števil, hkrati pa je skoraj povsod neničelna. Posledično bi morali pri filtriranju upoštevati vse vzorce signala, kar je v praksi zelo računsko zahtevno, poleg tega pa večina vzorcev le malo vpliva na končni rezultat rekonstrukcije.

- Pravilna rekonstrukcija je odvisna že od samega vzorčenja izvornega signala, ob nepravilnem zajemu podatkov namreč lahko po rekonstrukciji pride do neželenega popačenja (angl. aliasing). V praksi signali običajno niso frekvenčno omejeni, zato je pomembno, da jih že ob zajemu filtriramo z nizkoprepustnim filtrom.

V praksi se zato uporabljajo mnoge aproksimacije funkcije $h(t)$, najpogosteje funkcija najbližjega soseda (angl. nearest neighbor), trilinearna interpolacija in trikubična interpolacija. Funkcija najbližjega soseda in trilinearna interpolacija sta poleg idealnega filtra prikazani na sliki 2.4. Naštete funkcije se med seboj razlikujejo predvsem po številu upoštevanih vzorcev in času izračunavanja. V praksi je smiselna uporaba algoritma, ki na podlagi ocene relevantnosti podatkov uporabi primeren rekonstrukcijski filter. Ocena relevantnosti je lahko oddaljenost od kamere oz. fokusne razdalje, lokalna homogenost polja ali prosojnost materiala. V tem delu smo uporabili trilinearno interpolacijo, saj je rekonstruiran signal tako dovolj gladek za naše potrebe, hkrati pa je dovolj enostavno izračunljiv. Določene prednosti ima tudi pri iskanju ničel, kar je opisano v poglavju 2.4. Za prosojne materiale smo uporabili funkcijo relevantnosti, ki oddaljenosti od kamere oz. fokusne razdalje določi natančnost vzorčenja volumna. Opisana je v poglavju 2.7.1.

Posebne obravnave je deležna tudi rekonstrukcija lokalnega gradienta. Gradient potrebujemo za izračun normale nivojske ploskve v postopku senčenja in pri iskanju ničle implicitne funkcije z Newton-Raphsonovo metodo



Slika 2.4: Primerjava različnih rekonstrukcijskih filtrov.

(glej poglavje 2.4). Najenostavnejša tehnika ocenjevanja so centralne diference, ki kljub splošnemu slabemu mnenju (v primerjavi s 3D različico Sobelovega operatorja) dajejo presenetljivo dobre rezultate. Za vsako dimenzijo potrebujemo 2 vrednosti, torej 6 vrednosti za tri dimenzije v našem primeru. Če želimo tudi za normale uporabiti trilinearno interpolacijo, moramo gradient vzorčiti v 8 sosednjih točkah, nato pa pridobljene vrednosti interpolirati. Čeprav se na prvi pogled zdi, da je potrebnih 48 vzorcev funkcije, se to število zniža na 32, če upoštevamo, da se nekatere točke vzorčenja uporabijo večkrat. Kljub temu je to v primerjavi z 8 vzorci za vrednost funkcije še vedno veliko. Za računanje interpolacije vektorjev potrebujemo seveda tudi več računskega časa in pomnilnika kot za interpolacijo skalarjev. Za rekonstrukcijo gradienta obstajajo tudi boljše metode. Med najbolj popularnimi je 3D inačica Sobelovega operatorja.

2.4 Nivojske ploskve in reševanje implicitnih enačb

En izmed načinov vizualizacije volumetričnih podatkov je upodabljanje nivojskih ploskev. Nivojska množica (angl. level set) je v matematičnem smislu

množica L_μ točk $\mathbf{x}$, za katere velja

$$L_\mu = \{\mathbf{x} \mid \Phi(\mathbf{x}) = \mu\}. \tag{2.12}$$

Vrednost μ se imenuje nivojska konstanta. Ker gre v našem primeru za nivojske ploskve, velja dodatna omejitev $\mathbf{x} \in \mathbb{R}^3$. Z uporabo metode metanja žarkov se ta problem poenostavi v enodimenzionalni problem iskanja ničle:

$$\Psi(\lambda) \stackrel{\text{def.}}{=\!=} (\Phi \circ \mathbf{r})(\lambda) - \mu = 0. \tag{2.13}$$

Ker je funkcija Φ definirana implicitno, se moramo poslužiti numeričnih metod za iskanje ničel. Na tem področju je običajno najboljša izbira Newton-Raphsonova metoda[2], ki v našem primeru pomeni sledečo iteracijo:

$$\lambda_{k+1} = \lambda_k - \frac{\Psi(\lambda_k)}{\Psi'(\lambda_k)} \tag{2.14}$$

$$= \lambda_k - \frac{(\Phi \circ \mathbf{r})(\lambda_k) - \mu}{(\nabla\Phi \circ \mathbf{r})(\lambda_k) \cdot \mathbf{r}'(\lambda_k)} \tag{2.15}$$

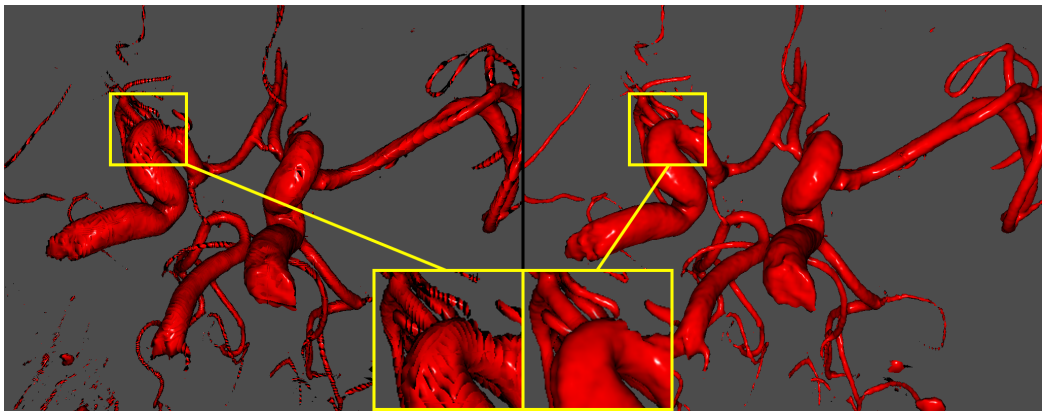
$$= \lambda_k - \frac{(\Phi \circ \mathbf{r})(\lambda_k) - \mu}{(\nabla\Phi \circ \mathbf{r})(\lambda_k) \cdot \hat{\mathbf{D}}(i,j)}. \tag{2.16}$$

Zaplete se pri številu potrebnih vzorčenj volumna (še posebej za oceno gradienta, glej poglavje 2.3) in pri nestabilnosti Newton-Raphsonove metode. V našem primeru je, tudi zaradi uporabe trilinearne interpolacije, primernejša metoda regula falsi[3]. Metoda je primerna takrat, ko zanesljivo vemo, da ničla leži na intervalu $[a, b]$. V tem primeru izračunamo ničlo c linearne interpolacije teh dveh točk. Na podlagi vrednosti funkcije v točki c se odločimo za nadaljevanje iteracije na intervalu $[a, c]$ oz. $[c, b]$. Iteracija, ki smo jo uporabili tudi v naši implementaciji, je naslednja:

$$c_k = b_k - \frac{\Psi(b_k)(b_k - a_k)}{\Psi(b_k) - \Psi(a_k)}. \tag{2.17}$$

[2] `http://en.wikipedia.org/wiki/Newton's_method`

[3] `http://en.wikipedia.org/wiki/Regula_falsi`



Slika 2.5: Primerjava izrisov z uporabo (levo) in brez uporabe (desno) metode regula falsi.

Vrednosti a_k in b_k, $a_k < b_k$, sta krajišči intervala, na katerem leži ničla. Vrednost c_k je naslednji približek, na podlagi katerega se kasneje odločimo za nadaljevanje iteracije v levem ($[a_k, c_k]$) ali desnem ($[c_k, b_k]$) intervalu. Metoda zanesljivo konvergira vsaj tako hitro kot navadna bisekcija, toda potencialno precej počasneje od Newton-Raphsonove. Vzorčenj funkcije je tudi precej manj, saj moramo za en korak iteracije le dvakrat vzorčiti signal - v nasprotju z Newton-Raphsonovo metodo, kjer imamo eno vzorčenje vrednosti funkcije in eno vzorčenje njenega gradienta. Ugotovili smo, da za naše potrebe zadoščajo 4 koraki metode regula falsi. Primerjava končnih slik z različnim številom korakov iteracije je na sliki 2.5.

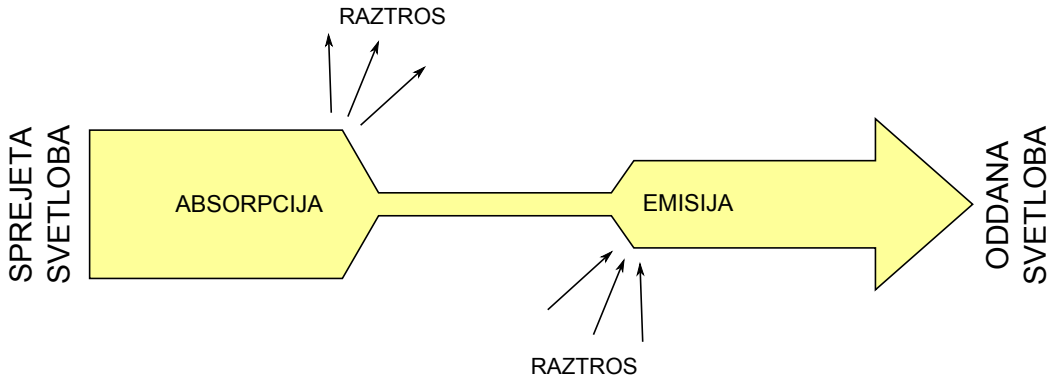
Težava opisanih numeričnih metod za iskanje ničel je v iskanju začetnega približka. Pri Newton-Raphsonovi metodi je končni rezultat iteracije močno odvisen od začetnega približka. Poleg nepredvidljivega vedenja metode lahko pride tudi do ciklične iteracije ali divergentnega zaporedja. Lahko naletimo celo na stacionarno točko, kjer je odvod enak 0 in korak iteracije torej nedefiniran. Do takšnih težav pri metodi regula falsi ne pride, a vseeno ima metoda nekaj pomanjkljivosti. Namesto začetne točke moramo najti začetni interval. Če na tem intervalu leži več ničel, ne moremo predvideti, katero ničlo bo metoda našla, zato je pomembno, da je začetni interval karseda na-

tančno definiran. To pa žal ni tako enostavna naloga - za učinkovito in hitro iskanje začetnega intervala se moramo namreč poslužiti naprednejših metod, ki zahtevajo tudi veliko predprocesiranja. Najpogostejši metodi za ta namen sta transformacija razdalj (angl. distance transform) in hierarhična predstavitev volumna. Oba pristopa omogočata učinkovito preskakovanje praznega prostora in posledično zelo hitro iskanje začetnih približkov. Transformacija razdalj se uporablja predvsem v algoritmu sphere tracing, hierarhične predstavitve pa so precej bolj splošne in lahko shranjujejo najrazličnejše podatke. Več o hierarhičnih predstavitvah je opisano v poglavju 2.6.

2.5 Optični modeli

Za verodostojno simulacijo svetlobe moramo najprej spoznati nekaj predpostavk in fizikalnih zakonov. Obstaja mnogo optičnih modelov, ki narekujejo simulacijo obnašanja svetlobe v volumnu. V nadaljevanju bomo predstavili emisijski, absorpcijski in emisijsko-absorpcijski model. Vsak od teh modelov je primeren za različne aplikacije, pogosto pa je emisijsko-absorpcijski model dovolj splošen, da z njim lahko upodobimo večino volumnov po naših željah. Za doseganje interaktivnosti upodabljanja se ti modeli omejijo na simuliranje le nekaterih lastnosti svetlobe. Kompleksnejši modeli, ki poleg neposredne svetlobe upoštevajo tudi sence in raztros (angl. scattering), so računsko prezahtevni, zato jih omenjamo le za namene pregleda področja. Poglobljen pregled najbolj popularnih optičnih modelov je podan v [24].

V vseh opisanih optičnih modelih predpostavljamo, da volumen sestoji iz infinitezimalno majhnih delcev, ki lahko absorbirajo, sevajo ali preusmerjajo svetlobo. Absorpcija in emisija se računata le v smeri proti kameri, preusmeritve pa upoštevajo tudi sosednje žarke. Rezultat absorpcije sprejete svetlobe je nižja intenziteta oddane svetlobe, rezultat emisije je višja intenziteta oddane svetlobe, preusmeritev pa se lahko kaže kot emisija ali kot absorpcija. Pri preusmeritvi se skupna energija žarkov ne spreminja. Vsi opisani dogodki se dogajajo hkrati in zvezno. Dogajanje v eni točki je shematsko prikazano



Slika 2.6: Teorija transporta svetlobe.

na sliki 2.6. V sledečih poglavjih v enačbah nastopa *intenziteta svetlobe*, ki jo označujemo z I. Količino lahko neposredno (z identitetno preslikavo) preslikamo v svetlost zaslonske točke. Če želimo modelirati tudi barve, jo obravnavamo kot vektorsko količino.

2.5.1 Emisijski model

Najenostavnejši optični model je emisijski model. Volumen v tem primeru sestoji iz infinitezimalno majhnih delcev, ki sevajo svetlobo. Naj bo $s \geq 0$ prepotovana pot žarka svetlobe. V vsaki točki s delci volumna dodajo svoj delež k intenziteti svetlobe. To lahko zapišemo z diferencialno enačbo:

$$\frac{dI(s)}{ds} = g(s), \tag{2.18}$$

katere rešitev je:

$$I(s) = I_0 + \int_0^s g(t)dt. \tag{2.19}$$

V zgornji enačbi je $g(s) \geq 0$ *emisijski koeficient*. Višji emisijski koeficient pomeni hitrejše naraščanje intenzitete svetlobe. Količina I_0 je svetlost ozadja in je v praksi pogosto enaka 0. V tem modelu predpostavljamo, da gostota volumna ne vpliva na intenziteto svetlobe. Na ta način lahko dokaj dobro

opišemo obnašanje svetlobe v ognju ali podobnih svetlobnih pojavih. Težava se pojavi pri upodabljanju, saj intenziteta svetlobe v modelu v nasprotju z barvami zaslona ni omejena. Tako lahko končna slika hitro postane povsem bela, saj svetlost hitro preseže maksimalno dovoljeno vrednost 1. Težavo lahko delno rešimo tako, da zmanjšamo vpliv emisijskega koeficienta na intenziteto svetlobe. Diferencialna enačba se po spremembi glasi:

$$\frac{dI(s)}{ds} = ag(s), \tag{2.20}$$

njena rešitev pa je:

$$I(s) = I_0 + a \int_0^s g(t)dt. \tag{2.21}$$

Koeficient a lahko spreminjamo tudi dinamično s spremljanjem maksimalne intenzitete svetlobe, projicirane na zaslon. Podoben učinek se v fotografiji imenuje HDRI (angl. high-dynamic-range imaging).

2.5.2 Absorpcijski model

Komplementaren model emisijskemu je absorpcijski model, saj emisijo volumna zanemarja, modelira pa le absorpcijo. Delci so še vedno infinitezimalno majhni, ampak so dovolj veliki, da lahko prestrezajo in absorbirajo svetlobne žarke. Fizikalno podlago za ta model predstavlja Beer-Lambertov zakon [4]. Model je primeren za modeliranje temnih snovi, na primer pepela ali premoga. Za potrebe modeliranja vpeljemo gostoto volumna $\kappa(s) \geq 0$, ki jo imenujemo tudi *absorpcijski koeficient*. Ta je neposredno odgovoren za hitrost padanja intenzitete svetlobe. Hkrati moramo upoštevati tudi intenziteto sprejete svetlobe, saj se absorpcija izraža kot njen delež. Diferencialna enačba tega optičnega modela se glasi:

$$\frac{dI(s)}{ds} = -\kappa(s)I(s), \tag{2.22}$$

z rešitvijo:

[4] `http://en.wikipedia.org/wiki/Beer-Lambert_law`

$$I(s) = I_0 \exp\left(-\int_0^s \kappa(t)dt\right). \tag{2.23}$$

V zgornji enačbi je I_0 svetlost ozadja in je v praksi pogosto enaka 1. Količina $\int_0^s \kappa(t)dt$ se imenuje tudi *optična globina*, za katero v literaturi pogosto zasledimo tudi oznako $\tau(s)$. Do točke s pride le delež $e^{-\tau(s)}$ intenzitete ozadja. Za potrebe upodabljanja se ta delež običajno preslika v kanal α barvnega modela RGBA s preslikavo $\alpha(s) = 1 - e^{-\tau(s)}$.

2.5.3 Emisijsko-absorpcijski model

Emisijsko-absorpcijski model je združitev emisijskega in absorpcijskega modela. V tem primeru moramo v diferencialni enačbi upoštevati obe zgoraj omenjeni optični lastnosti $g(s)$ in $\kappa(s)$. Model je primeren za marsikatero aplikacijo, tudi za interaktivne, saj je še vedno dovolj enostaven da omogoča učinkovite metode upodabljanja. Združena diferencialna enačba je tokrat:

$$\frac{dI(s)}{ds} = g(s) - \kappa(s)I(s), \tag{2.24}$$

njena rešitev pa:

$$I(s) = I_0 \exp\left(-\int_0^s \kappa(u)du\right) + \int_0^s g(t) \exp\left(-\int_t^s \kappa(u)du\right) dt. \tag{2.25}$$

Zadnja enačba se ponaša z imenom *integral volumetričnega upodabljanja* in je temelj za vizualizacijo volumetričnih podatkov. Vse metode, opisane v poglavju 1.2, na bolj ali manj neposreden način vrednotijo ta integral - metanje žarkov je le "najbolj neposredna" metoda, saj povsem ustreza konceptu svetobnega žarka, ki je temeljni konstrukt v opisanih optičnih modelih. Integral volumetričnega upodabljanja je v splošnem analitično nerešljiv, toda za njegovo reševanje obstajajo učinkovite numerične metode. Najpogostejša je rešitev z Riemannovo vsoto, ki smo jo uporabili tudi v naši implementaciji, obstajajo pa tudi boljši načini (trapezna ali simpsonova formula), ki integral ovrednotijo precej bolj natančno na račun daljšega procesiranja.

Poenostavljena enačba, ki upošteva le obnašanje svetlobe na površinah, se imenuje *integral upodabljanja*. Emisija in absorpcija znotraj materialov sta tokrat zanemarjeni, ostane pa člen za simuliranje odbojev svetlobe (BRDF). Tudi taka poenostavljena različica je analitično v splošnem nerešljiva, zato znani algoritmi s področja upodabljanja ploskev, na primer algoritem izsevnosti, MLT (angl. Metropolis light transport) in sledenje svetlobi, poskušajo integral numerično aproksimirati.

2.5.4 Sinteza slike

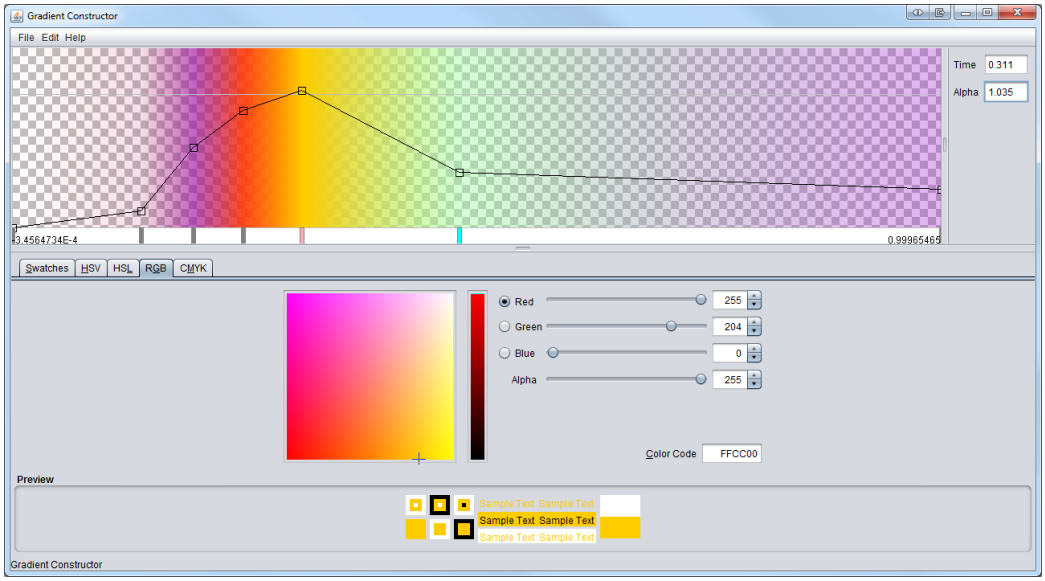
Medtem ko je sinteza slike pri upodabljanju nivojskih ploskev razmeroma enostavna, je ta postopek nekoliko kompleksnejši pri upodabljanju volumna s prosojnostjo. V tem primeru gre namreč za sestavljanje barv, ki jih pridobimo z Riemannovo vsoto. V integralu nastopajo tudi različne optične lastnosti materialov, na primer absorpcijski in emisijski koeficient, ki v vektorski obliki predstavljata barvo in prosojnost. Te lastnosti niso shranjene v volumnu, temveč jih pridobimo v t.i. postopku klasifikacije. Za potrebe klasifikacije smo razvili tudi interaktivno aplikacijo, ki nam omogoča lažje določanje preslikave med vrednostmi volumna in optičnimi lastnostmi. Prikazana je na sliki 2.7. Rezultati uporabe različnih klasifikacijskih gradientov so prikazani na sliki 2.8.

V nadaljevanju je predstavljena diskretizacija integrala. Za enostavnejši zapis bomo uporabljali $\kappa(u) = (\kappa \circ \Phi \circ \mathbf{r})(u)$ in $g(u) = (g \circ \Phi \circ \mathbf{r})(u)$, kar predstavlja klasifikacijski postopek. Število potrebnih vzorcev bomo označili z $n = \left\lfloor \frac{s}{\Delta u} \right\rfloor$.

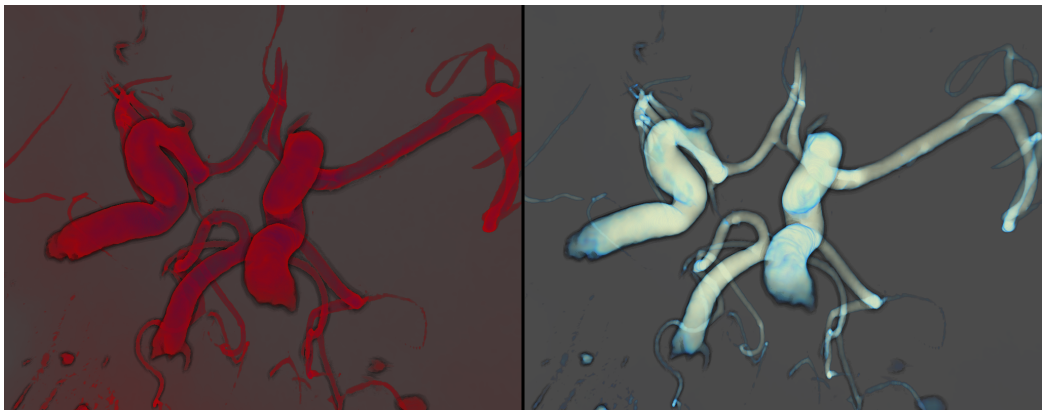
Optično globino lahko zapišemo kot

$$\exp\left(-\int_0^s \kappa(u)du\right) \approx \exp\left(-\sum_{i=0}^{n} \kappa(i\Delta u)\Delta u\right) = \prod_{i=0}^{n} e^{-\kappa(i\Delta u)\Delta u}. \quad (2.26)$$

Podobno lahko storimo z delno optično globino, prisotno znotraj drugega integrala:



Slika 2.7: Klasifikacijsko orodje Gradient Constructor.



Slika 2.8: Emisijsko-absorpcijski model z uporabo različnih klasifikacijskih gradientov.

$$\exp\left(-\int_t^s \kappa(u)du\right) \approx \exp\left(-\sum_{j=i+1}^{n} \kappa(i\Delta u)\Delta u\right) = \prod_{j=i+1}^{n} e^{-\kappa(i\Delta u)\Delta u}. \quad (2.27)$$

Vrednost i je v tem primeru števec zunanje vsote. Integral volumetričnega upodabljanja je z uporabo Riemannove vsote torej enak

$$I(s) = I_0 \exp\left(-\int_0^s \kappa(u)du\right) + \int_0^s g(t) \exp\left(-\int_t^s \kappa(u)du\right) dt \quad (2.28)$$

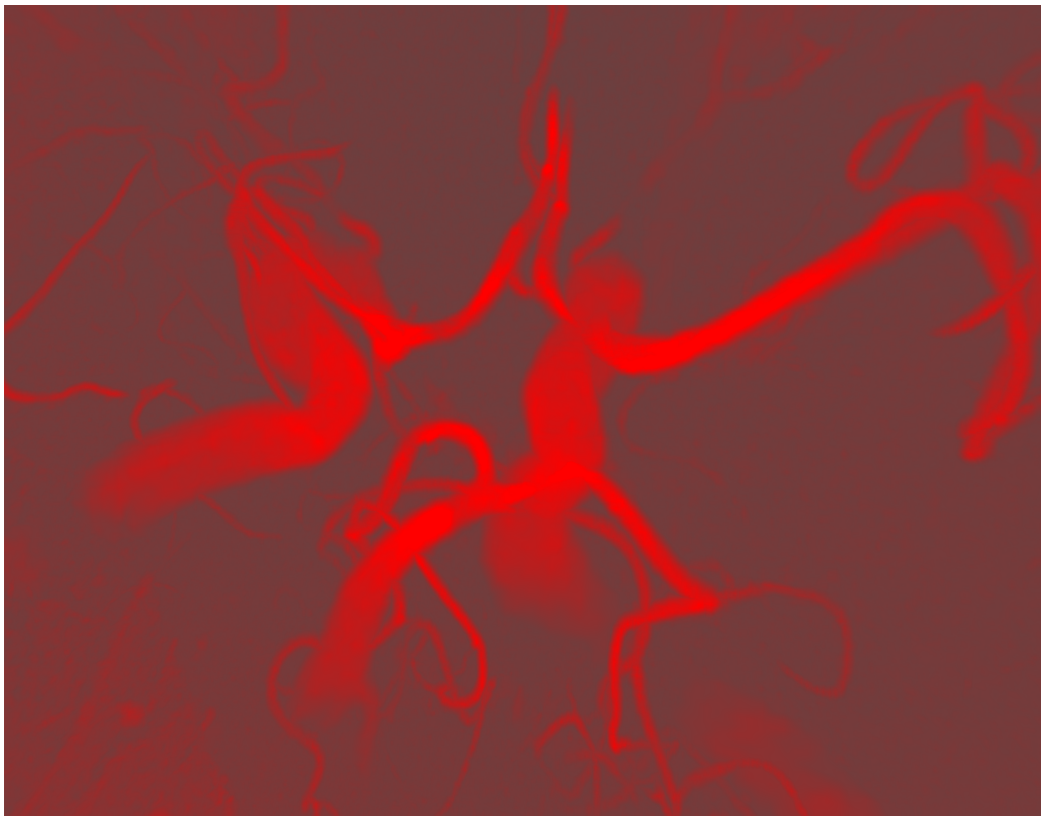
$$\approx I_0 \prod_{i=0}^{n} e^{-\kappa(i\Delta u)\Delta u} + \sum_{i=0}^{n} g(i\Delta t)\Delta t \prod_{j=i+1}^{n} e^{-\kappa(i\Delta u)\Delta u}. \quad (2.29)$$

Zadnja formula predstavlja sestavljanje barv s prosojnostjo, opisano v [34]. V naši aplikaciji je implementirano sestavljanje od spredaj nazaj, ki se dobro prilega algoritmu metanja žarkov, hkrati pa še vedno dovoljuje uporabo optimizacijskih struktur. Podobno kot barve se sestavlja tudi globina za globinsko sliko, ki se uporablja pri nekaterih vizualnih efektih, opisanih v poglavju 2.8.

2.5.5 Projekcija največje intenzitete

Zaradi svoje popularnosti v vizualizacijskih aplikacijah smo implementirali še eno metodo. Metoda projekcije največje intenzitete ali MIP (angl. maximum intensity projection), opisana v [27], je med najenostavnejšimi in najbolj razširjenimi metodami vizualizacije, čeprav je le specifičen primer emisijsko-absorpcijskega modela. Gre za iskanje največje vrednosti volumna in projiciranje te vrednosti na zaslon. Ker svetlost zaslonske točke odraža intenziteto prejete svetlobe, lahko za generiranje barve uporabimo kar linearno interpolacijo med črno in belo barvo (kombinacija z rdečo barvo je prikazana na sliki 2.9). Matematično to lahko zapišemo kot:

$$I = \max_{\lambda} (\Phi \circ \mathbf{r})(\lambda). \quad (2.30)$$



Slika 2.9: Projekcija največje intenzitete.

Zaradi svoje enostavnosti je metoda izredno hitra. Hitrost upodabljanja lahko bistveno povečamo z uporabo kakšne podatkovne strukture, na primer osmiškega drevesa, ki v vozliščih drži minimalno in maksimalno vrednost poddrevesa. Maksimalno vrednost lahko z uporabo osmiškega drevesa enostavno poiščemo tako, da se vedno spustimo v tisto poddrevo, katerega maksimalna vrednost je največja med poddrevesi, ki jih žarek svetlobe seka. Nato po potrebi pregledamo še ostala poddrevesa, ki jih žarek svetlobe prav tako seka. Ker v naši implementaciji osmiškega drevesa ne gradimo do najnižjega nivoja, moramo najnižje poddrevo preiskati linearno. V najslabšem primeru bo algoritem pregledal vsa poddrevesa, toda v praksi je to malo verjetno.

Kljub izredno hitri vizualizaciji pa ima metoda veliko pomanjkljivosti. Če uporabljamo ortografsko projekcijo, je na statični sliki nemogoče sklepati

o orientaciji volumna, saj je projekcija z nasprotne strani volumna vedno le zrcalna slika. Nemogoče je tudi sklepati o globini in medsebojnih razdaljah v prostoru. Poleg tega je v tej metodi odsotno kakršno koli senčenje, saj bi to popolnoma razbilo namen metode. Posledično se ob projekciji izgubijo vse topološke informacije. Prav tako iz istega razloga ne moremo uporabljati vizualnih efektov, opisanih v poglavju 2.8. Te težave do neke mere rešuje algoritem LMIP (angl. local maximum intensity projection) [38, 19], ki projicira prvi lokalni maksimum, ki sega nad neko mejo μ. Vrednost μ lahko uporabimo tudi kot prag pri projiciranju največje intenzitete, ko je le-ta prenizka, saj je pri nizkih vrednostih dobro viden vpliv šuma v zajeti sliki. V tem primeru ne izrišemo ničesar. Dojemanje orientacije, globine in medsebojnih razdalj v prostoru lahko bistveno izboljšamo z animacijo in perspektivno projekcijo.

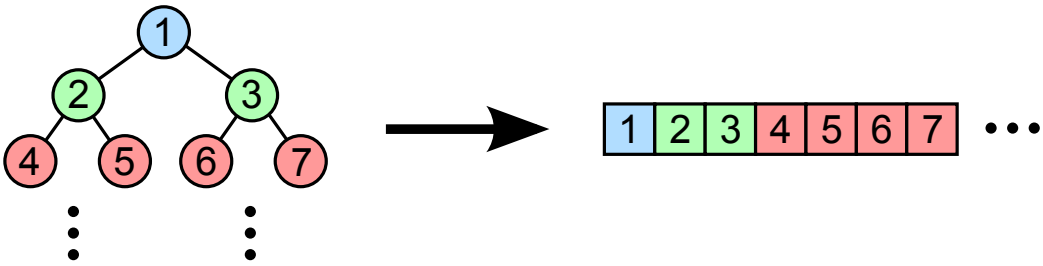
2.6 Osmiško drevo

V angiografskih volumetričnih slikah je veliko prostora praznega oz. homogenega, zato je smiselno, da taka območja v algoritmih hitro prepoznavamo in procesiramo. Učinkovito preskakovanje praznega in homogenega prostora je zato med najbolj učinkovitimi optimizacijami algoritma metanja žarkov. Vzorčenje takega prostora je namreč v večini primerov povsem nepotrebno, zato je cilj optimizacije zmanjšati število vzorčenj. Najpreprostejša in najpogostejša rešitev tega problema je uporaba hierarhične predstavitve volumna. Tu srečamo več pristopov, toda med najbolj popularnimi sta osmiško drevo in k-d drevo. Drevesi sta si zelo podobni, razlikujeta se le po razvejanosti in regularnosti deljenja prostora. K-d drevo je binarno, delitev prostora je neregularna, medtem ko ima vsako vozlišče osmiškega drevesa osem otrok, ki podprostor vedno razdelijo na osem enakih delov. Za oba pristopa obstaja mnogo različnih tehnik sprehajanja po drevesni strukturi in shranjevanja podatkov. Pregled tovrstnih metod najdemo v [36], priredbe in optimizacije za grafične procesorje pa v [5].

Pri shranjevanju podatkov imamo na voljo polno drevo ali redko drevo. V prvem primeru so vedno prisotni vsi otroci, medtem ko pri slednjem to ni nujno. Redka drevesa v splošnem porabijo več pomnilnika za shranjevanje metapodatkov o drevesu (kazalci na bloke podatkov, kazalci na otroke, maske prisotnosti otrok), toda pri dovolj homogenih oz. praznih volumnih so prihranki na ta račun veliki. Ker je struktura večinoma neodvisna od razporeditve podatkov v pomnilniku, je ta metoda tudi brez večjih sprememb v algoritmu pripravljena na sprehode, pri katerih dele drevesa po potrebi nalagamo z diska. Po drugi strani je polno drevo lepo strukturirano in v pomnilnik ga lahko zapišemo strnjeno. Poizvedbe in sprehodi so zaradi regularne strukture mnogo hitrejši. Za podatke z večjo entropijo je ta pristop zaradi manjše količine metapodatkov tudi bolj učinkovit v smislu porabljenega pomnilnika. Ker je bila interaktivnost aplikacije naš primarni cilj, smo se seveda odločili za polno osmiško drevo. Odločili smo se za popularen pristop, kjer so v vozliščih drevesa shranjene minimalne, maksimalne in povprečne vrednosti podatkov v pripadajočih poddrevesih. To nam omogoča hitro prepoznavanje praznih in homogenih območij, kar lahko s pridom uporabimo za pohitritev algoritma metanja žarkov.

Osmiško drevo smo v naši aplikaciji strukturirali tako, da so vozližča enega nivoja vedno zaporedno shranjena v pomnilniku. Shematski prikaz pomnilniške strukture je na sliki 2.10. Ker smo uporabili polno drevo, nam tak pristop olajša poizvedbe po drevesu, saj lahko položaje vozlišč v pomnilniku enostavno izračunamo. Indeks i-tega otroka vozlišča z indeksom k lahko namreč izredno hitro pridobimo z enim množenjem in enim seštevanjem po formuli $8k + i$.

Daljšega premisleka je bila deležna sama struktura drevesa. Predvsem smo morali paziti na količino pomnilnika, ki ga struktura porabi. Če predpostavljamo, da so volumetrični podatki dimenzij $(2^n)^3$ in tako vsak podatek volumna kot vsako vozlišče drevesa porabita eno enoto pomnilnika, imamo na najnižjem nivoju $(2^n)^3$ podatkov, na nivoju višje $\frac{(2^n)^3}{8} = 2^n$ podatkov, na nivoju višje spet le eno osmino tega in tako vse do vrha drevesa, kjer je le en



Slika 2.10: Shematični prikaz strukture osmiškega drevesa v pomnilniku.

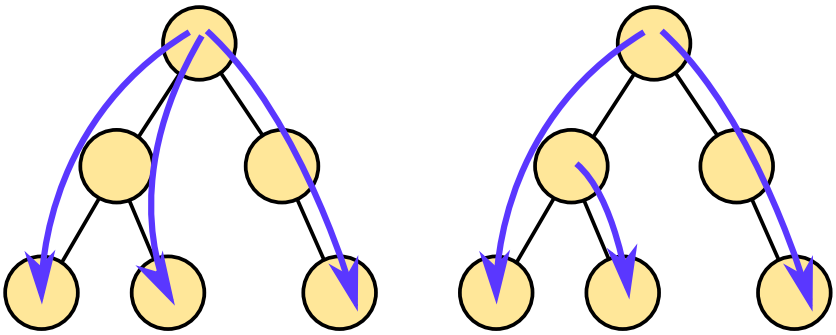
podatek. Če količino podatkov seštejemo, dobimo geometrijsko vrsto

$$\sum_{k=1}^{\infty} \left(\frac{1}{8}\right)^k = \frac{1}{1-\frac{1}{8}} - 1 = \frac{1}{7} \approx 0.1429, \tag{2.31}$$

kar pomeni, da drevesna struktura porabi le okrog 14% dodatnega prostora. Realnost pa žal ni tako lepa. Volumetrični podatki so namreč 32-bitna števila v plavajoči vejici (float), vozlišča drevesa pa po naših zahtevah shranjujejo tri taka števila. Posledično bi drevesna struktura zasedla okrog 43% dodatnega prostora. Dodatna težava se pojavi z upoštevanjem dimenzij volumna. Ker želimo polno drevo, ga moramo zgraditi tako, da so vedno prisotni vsi otroci. Posledično moramo drevesno strukturo (ne pa tudi podatkov!) razširiti do dimenzij $(2^n)^3$, kar v najslabšem primeru poveča porabo pomnilnika za faktor 8. To bi torej pomenilo že dodatnih 343% prostora, kar pa je seveda nesprejemljivo. Zato smo se odločili, da drevesa ne bomo gradili do najnižjega nivoja, temveč se bomo ustavili že tri nivoje višje. Če nato seštejemo število vozlišč, ugotovimo, da zasedajo le

$$\sum_{k=3}^{\infty} \left(\frac{1}{8}\right)^k = \frac{1}{448} \approx 0.002232 \tag{2.32}$$

dodatnega pomnilniškega prostora. Če upoštevamo, da se ta številka zaradi razširjanja volumna in večjega števila podatkov v vsakem vozlišču drevesa poveča za faktor 24, to vseeno nanese le dobrih 5% dodatnega prostora. Performančno gledano pomanjkanje ogromnega dela drevesne strukture ni



Slika 2.11: Shema delovanja algoritma kd-restart (levo) in kd-backtrack (desno). Algoritem kd-backtrack si zapomni skupno starševsko vozlišče.

tako hud udarec, saj že sam sprehod po drevesu porabi nekaj dragocenega časa, kar smo v naši implementaciji kompenzirali s pogostejšim vzorčenjem volumna v listih drevesa.

Za sprehod po drevesu smo imeli več možnosti. V najnovejših aplikacijah se pogosto uporabljajo skladovni algoritmi, ki minimizirajo število obiskov vozlišč drevesa. Primer uporabe takega algoritma za upodabljanje volumnov je v [14], kjer je implementiran algoritem [35]. Tovrstni pristopi zahtevajo implementacijo sklada, kar na grafičnih karticah ni trivialno, poleg tega pa so le sodobni grafični procesorji sposobni izvajati kontrolne ukaze, ki so nujno potrebni pri teh metodah. Kljub številnejšim poizvedbam in nekoliko redundantnem sprehodu se zaradi učinkovitejših implementacij uporabljata algoritma kd-restart in kd-backtrack, predstavljena v [5]. Zaradi izsledkov testiranj, predstavljenih v [36], kjer so skladovni algoritmi prav iz prej omenjenih razlogov počasnejši, smo se odločili za najenostavnejši algoritem kd-restart.

Algoritem kd-restart ob vsaki novi poizvedbi ponovno začne sprehod v korenskem vozlišču. Za sprehod v globino uporablja referenčno točko, ki v danem vozlišču vodi sprehod v otroško vozlišče, ki vsebuje to točko. Med vsakim obiskom se algoritem odloča ali bo sprehod nadaljeval v trenutni veji ali pa bo referenčno točko prestavil drugam. Algoritem se zaključi, ko izstopimo iz korenskega vozlišča ali ko z žarkom dosežemo nivojsko ploskev.

Razlika med algoritmoma kd-restart in kd-backtrack je v obnašanju pri spremembi referenčne točke. Algoritem kd-backtrack si namreč zapomni skupno starševsko vozlišče in sprehod po drevesu začne od tega vozlišča dalje, medtem ko kd-restart v vsakem primeru začne znova v korenskem vozlišču. Razlika med algoritmoma je na enostavnem binarnem drevesu shematsko prikazana na sliki 2.11. V teoriji je kd-backtrack precej hitrejši, ko je drevo globoko in ko je sprememba referenčne točke majhna (kar se v resnici pogosto dogaja pri metanju žarkov). Takrat namreč lahko sprehod začne zelo nizko v drevesu, kd-restart pa mora začeti v korenskem vozlišču. Zaradi manjše količine potrebne logike je kd-restart vseeno dovolj hiter za interaktivne aplikacije.

2.7 Adaptivne metode

2.7.1 Adaptivno vzorčenje

Adaptivno vzorčenje je način optimizacije hitrosti, ki se pogosto uporablja pri upodabljanju volumetričnih podatkov s prosojnostjo. Ko nad podatki ni zgrajena nobena optimizacijska podatkovna struktura, lahko čas upodabljanja vseeno skrajšamo tako, da z ocenjevanjem lokalne homogenosti oz. pomembnosti območja redkeje vzorčimo signal. Adaptivno vzorčenje je še posebej uporabno, ko integral upodabljanja vrednotimo z metodo Monte-Carlo. Hitrost konvergence metode Monte-Carlo se namreč izredno poveča z uporabo *a priori* informacij o volumnu. Običajno informacije o volumnu pridobivamo tekom vzorčenja, hkrati pa s pridobljenimi informacijami dinamično prilagajamo območja vzorčenja, zato pravimo, da vzorčimo adaptivno.

V naši implementaciji smo preizkusili dva adaptivna pristopa za vzorčenje. Prvi uporablja informacije osmiškega drevesa za preskakovanje praznega in homogenega prostora, drugi pa hitrost vzorčenja spreminja glede na oddaljenost od kamere oz. od fokusne razdalje. Osmiško drevo shranjuje minimalne, maksimalne in povprečne vrednosti volumna, ki smo jih s pridom izkoristili za ocenjevanje homogenosti. Za oceno smo vzeli razliko med

maksimalno in minimalno vrednostjo. Ko je ta razlika manjša od neke predefinirane vrednosti, algoritem brez vzorčenja oceni vrednost poddrevesa s povprečno vrednostjo, ki jo prebere iz trenutnega vozlišča drevesne strukture.

Drug adaptivni pristop temelji na oddaljenosti od kamere oz. fokusne razdalje. V ta namen smo konstruirali funkcijo, ki glede na globino žarka vrne hitrost vzorčenja:

$$s(\lambda) = ((e^{-k_1\lambda^2} + e^{-k_2(\lambda-\lambda_0)^2})(s_1 - s_0) + s_0)e^{-k_3\lambda^2}, \tag{2.33}$$

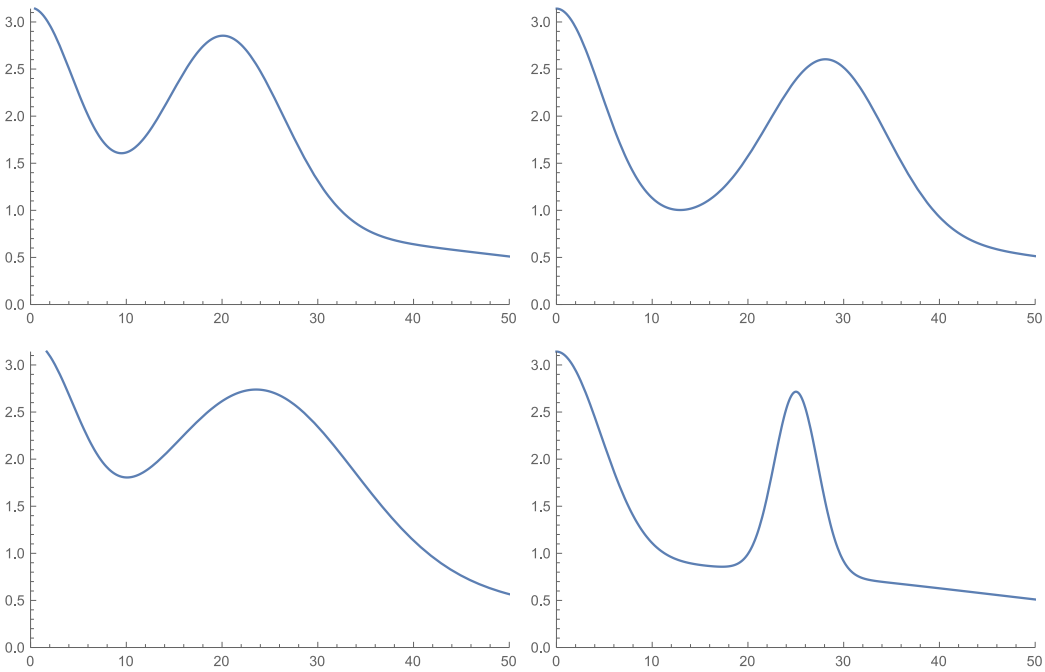
kjer so uporabljeni naslednji parametri:

- λ je prosti parameter žarka, ki ob normaliziranem smernem vektorju predstavlja globino,
- λ_0 je fokusna razdalja,
- k_1, k_2 in k_3 določajo širino Gaussovih funkcij, ki jih uporabljamo za povečanje hitrosti vzorčenja,
- s_0 in s_1 predstavljata okvirno minimalno in maksimalno histrost vzorčenja.

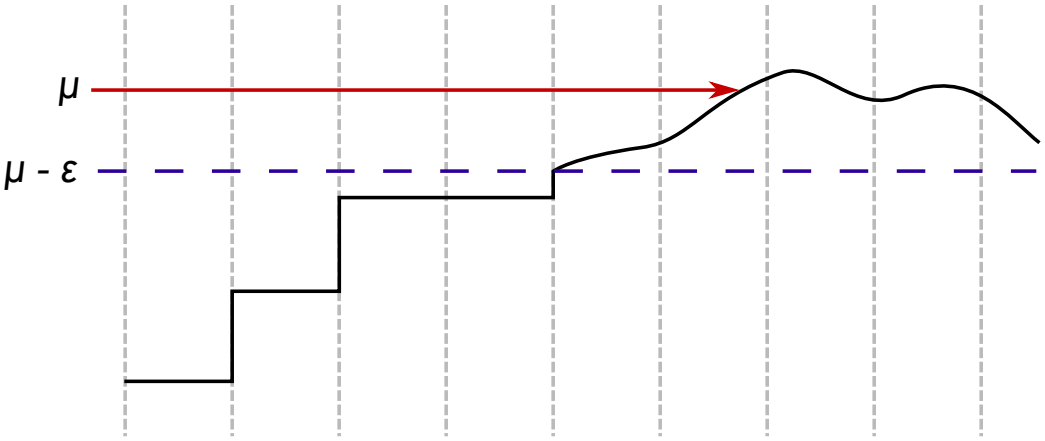
Prvi dve Gaussovi funkciji povečata natančnost vzorčenja neposredno pred kamero in v bližini fokusne razdalje, zadnja Gaussova funkcija pa poskrbi za padanje natančnosti vzorčenja pri velikih razdaljah od kamere. Nekaj funkcij za različne vrednosti parametrov je prikazanih na sliki 2.12. Za lažje izbiranje vrednosti parametrov smo razvili tudi interaktivni pripomoček.

2.7.2 Adaptivna rekonstrukcija

Tehnika, ki se pogosto uporablja za hitrejše vzorčenje signala v praznih, homogenih ali nerelevantnih območjih, je adaptivna rekonstrukcija. Ker običajno nimamo potrebe po natančni rekonstrukciji signala v takih predelih volumna, lahko privzeto vzorčimo z enostavnejšim rekonstrukcijskim



Slika 2.12: Različni grafi adaptivne funkcije vzorčenja. Vertikalna os predstavlja hitrost vzorčenja v številu vzorčenj vzdolž žarka na enoto (vzorec) volumna, horizontalna os pa globino žarka. V zgornji vrsti je prikazana sprememba parametra λ_0, v spodnji pa sprememba parametra k_2.

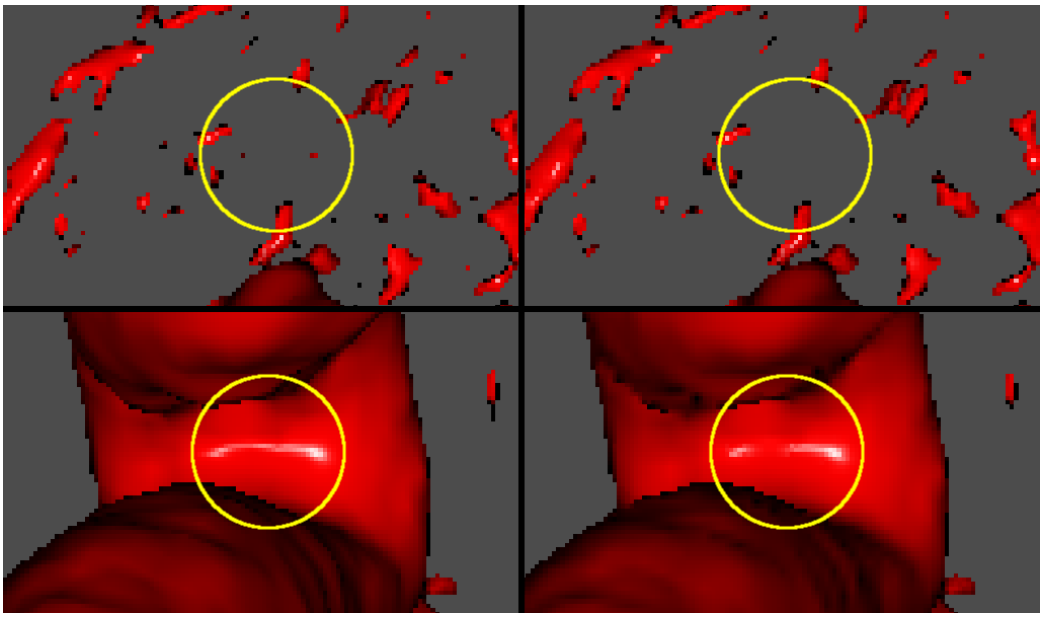


Slika 2.13: Shematičen prikaz adaptivne rekonstrukcije.

filtrom, nato pa natančnost vzorčenja po potrebi povečamo z zamenjavo rekonstrukcijskega filtra. Tehnika se je izkazala za učinkovito pri upodabljanju nivojskih ploskev, kjer večino volumna predstavlja prazen prostor. Trilinearno interpolacijo smo v tem primeru vključili le v bližini nivojske ploskve, ki smo jo zaznali z oddaljenostjo ε vzorčene vrednosti od dane nivojske konstante μ. Koncept je prikazan na sliki 2.13. Adaptivna rekonstrukcija v tem primeru ni vplivala na kvaliteto končne slike, hitrost upodabljanja pa se je bistveno povečala.

2.7.3 Redko metanje žarkov

Redko metanje žarkov je način zmanjšanja števila primarnih žarkov, ki jih pošljemo skozi volumen. Gre za način izkoriščanja koherence končne slike, saj se le-ta le malo spreminja med sosednjimi zaslonskimi točkami. Splošna različica metode je predstavljena v [18], njena izboljšava pa v [12]. Bistvo metode je prilagajanje števila žarkov glede na oceno homogenosti območja na končni sliki. Ker so metode, predstavljene v omenjenih člankih, prilagojene za izvajanje na centralni procesni enoti, smo v naši implementaciji metodo poenostavili tako, da se izvede le ena iteracija izboljšave slike. V prvem prehodu upodabljanja se pošljejo žarki le za vsako drugo zaslonsko točko, torej



Slika 2.14: Primerjava izrisov brez uporabe (levo) in z uporabo (desno) metode redkega metanja žarkov.

za vsako točko (i, j), kjer sta i in j sodi števili. V drugem prehodu upodabljanja se vrednosti vmesnih zaslonskih točk interpolirajo. Absolutne razlike med vrednostmi sosednjih zaslonskih točk se uporabijo za oceno homogenosti slike. Če so te razlike večje od neke preddefinirane vrednosti, se interpolirana vrednost zamenja z metanjem žarka. Tako kot adaptivna rekonstrukcija se je tudi redko metanje žarkov izkazalo za zelo učinkovito, saj močno zmanjša število potrebnih žarkov - ozko grlo te metode upodabljanja. Slaba stran metode je nezmožnost upodabljanja predmetov, ki so na končni sliki ožji od dveh zaslonskih točk. Take elemente namreč primarni žarki v večini primerov zgrešijo, česar pa zaradi načina ocenjevanja homogenosti slike ne moremo zaznati (nekaj primerov je na sliki 2.14). Kljub temu so taki elementi redko relevantni in jih je možno po potrebi izrisati ponovno po spremembi položaja kamere.

2.8 Vizualni efekti

Računalniška grafika v igrah pogosto išče kompromise med kvaliteto posebnih vizualnih efektov in verodostojno simulacijo obnašanja svetlobe. V računalniških igrah je namreč hitrost vizualizacije kritičnega pomena. Ker je bil tudi naš primarni cilj hitra in kvalitetna vizualizacija, smo se odločili, da nekaj vizualnih efektov preizkusimo tudi na področju medicinske vizualizacije.

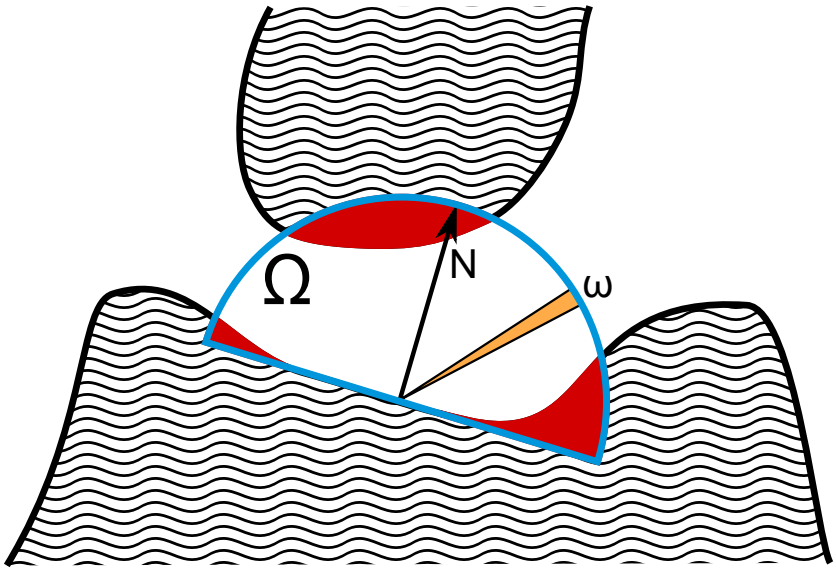
Vizualni efekti se praviloma računajo na zaslonski sliki. Pravimo, da so efekti *slikovno-prostorski*. Tudi odloženo upodabljanje, opisano v [42], bi lahko označili za množico vizualnih efektov. Pri večini popularnih efektov gre za enostavno konvolucijo slik s filtrom, toda to ni nujno. Pri mnogih efektih, predvsem tistih za odloženo upodabljanje, so namreč izračuni precej bolj kompleksni. Popularni in v široki uporabi so na primer korekcija svetlosti in kontrasta, gama korekcija, zameglitveni filter, preslikava barv (angl. tone mapping), iskanje robov in še mnogi drugi. Na področju računalniških iger glede na popularnost izstopajo zameglitveni filtri, HDRI in iskanje robov, med bolj kompleksnimi pa prevladujeta slikovno-prostorsko ambientno zastiranje (angl. screen-space ambient occlusion - SSAO) in globinska ostrina (angl. depth of field - DOF). Efekta poskušata poudariti topološke in globinske informacije, ki jih volumetrična slika vsebuje, z grobo simulacijo globalne osvetlitve oz. s simulacijo leče.

2.8.1 Ambientno zastiranje

Pri metodi ambientnega zastiranja gre za ocenjevanje dostopnosti površine. Lokalna dostopnost je definirana kot verjetnost, da žarek v polsferi okrog normale nad površino preseka površino predmeta. To lahko zapišemo kot:

$$A_{\mathbf{p}} = \frac{1}{\pi} \int_{\Omega} V_{\mathbf{p},\omega} (\hat{N} \cdot \hat{\omega}) d\omega, \tag{2.34}$$

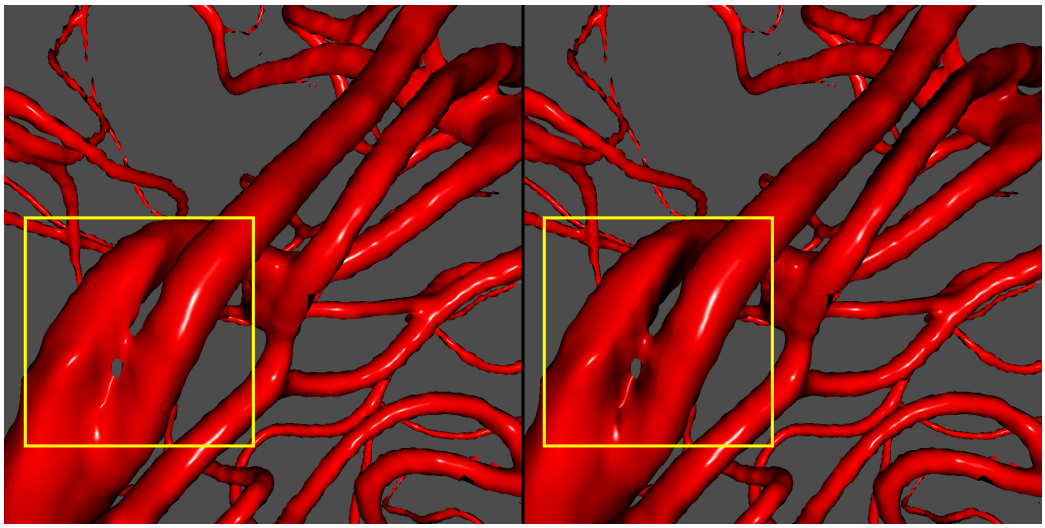
kjer je $A_{\mathbf{p}}$ lokalna dostopnost v točki $\mathbf{p}$ na površini, Ω polsfera okrog normale $\hat{N}$ nad površino in $V_{\mathbf{p},\omega}$ funkcija dostopnosti točke $\mathbf{p}$ s prostorskega kota



Slika 2.15: Shematičen prikaz metode ambientnega zastiranja. Rdeča področja povzročajo močnejše zastiranje.

ω. Omenjeni parametri so shematsko prikazani na sliki 2.15. Poleg funkcije dostopnosti je tu bistvenega pomena velikost polsfere, saj v formuli vpliv razdalje ni opredeljen. Funkcija dostopnosti je pogosto le preprosta indikatorska funkcija, ki je enaka 1, če je točka iz dane smeri dostopna, v nasprotnem primeru pa je enaka 0. Pri simulacijah prosojnih materialov je lahko tudi bolj kompleksna. V naši implementaciji smo lokalno dostopnost aproksimirali z metodo Monte-Carlo. Ko algoritem določi točko presečišča primarnega žarka z volumnom, v polsfero nad površino vrže še določeno število kratkih žarkov, ki služijo vrednotenju funkcije dostopnosti. Funkcija v naši implementaciji vrne vrednost 1, če je vrednost vzorca signala v obravnavani točki večja ali enaka nivojski konstanti μ.

Pridobljena vrednost se uporabi pri senčenju površin, kjer se intenziteta svetlobe pomnoži še s faktorjem ambientnega zastiranja $1 - A_{\mathbf{p}}$. S tem dosežemo, da so bolj zakrite površine temnejše. Rezultat je prikazan na sliki 2.16. Dokazano je, da metoda ambientnega zastiranja izboljša dojemanje globine in medsebojnih položajev v prostoru [15], toda zaradi temnejših pre-



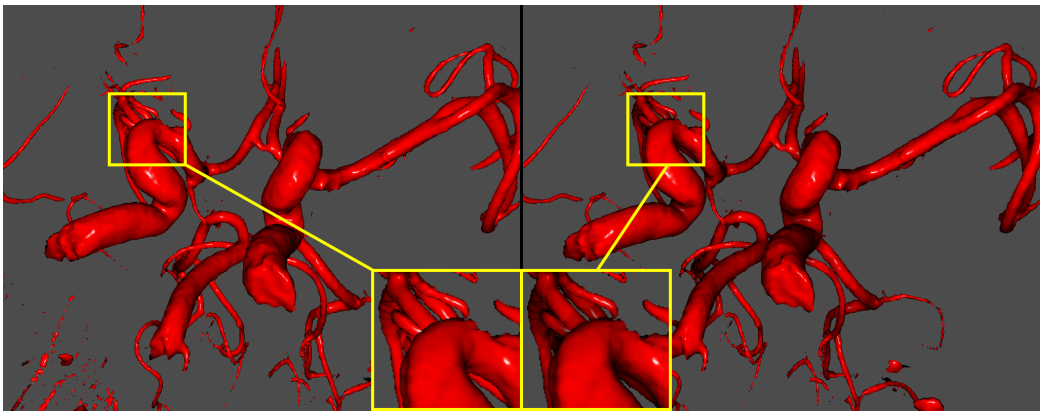
Slika 2.16: Primerjava izrisov brez uporabe (levo) in z uporabo (desno) metode ambientnega zastiranja.

delov slike lahko opazovalec spregleda določene bistvene elemente volumna, ki bi v primeru angiografije lahko bili odločilnega pomena v diagnostičnem procesu.

Opisan postopek daje dobre rezultate pri upodabljanju nivojskih ploskev, za prosojne materiale pa je tak model dostopnosti žal preveč poenostavljen. Boljši modeli za upodabljanje volumnov namreč upoštevajo tudi optično globino žarkov dostopnosti. V praksi se v ta namen pogosto uporablja naslednja formula:

$$I(\mathbf{p}) = \int_a^{R_\Omega} \frac{w}{R_\Omega - a} \exp\left(-\int_a^s \kappa(u)du\right) ds, \tag{2.35}$$

kjer je $I(\mathbf{p})$ intenziteta vpadne svetlobe v točki $\mathbf{p}$, w utež, s katero lahko do neke mere simuliramo usmerjeno zastiranje ter R_Ω in a veliki in mali radij sfere okrog točke $\mathbf{p}$. Zadnji dve vrednosti sta potrebni za lokalnost efekta. Vrednost a je ponavadi le majhen odmik, ki točki preprečuje prekrivanje same sebe. Zaradi dokaj kompleksne aproksimacije se ta integral kljub dobrim rezultatom le redko uporablja v interaktivnih aplikacijah.

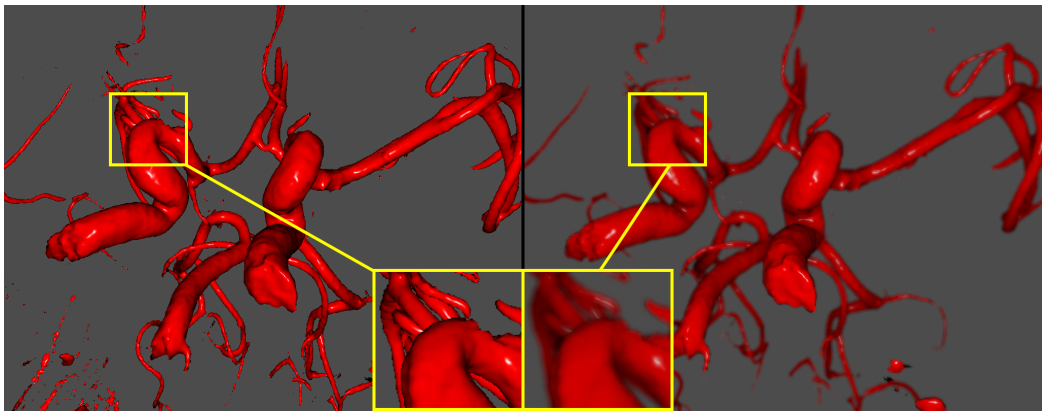


Slika 2.17: Primerjava izrisov brez uporabe (levo) in z uporabo (desno) efekta slikovno-prostorskega ambientnega zastiranja.

Kljub očitni kompleksnosti algoritmov za aproksimacijo ambientnega zastiranja obstaja privlačna alternativa, ki se izvaja v slikovnem prostoru. Slikovno-prostorsko ambientno zastiranje je tehnika, ki so jo razvili leta 2007 v podjetju Crytek za svoj grafični pogon. V tem pristopu se iz položaja zaslonske točke in globinske slike rekonstruira položaj v prostoru, nato pa se oceni dostopnost na podoben način, kot je opisano zgoraj. Seveda obstajajo tudi izboljšave osnovnega algoritma, ki upoštevajo tudi sliko normal. Ta je običajno na voljo, če uporabljamo odloženo upodabljanje. Naša implementacija temelji na rahlo prilagojeni različici Crytekovega algoritma[5].

Algoritem vzorči globinsko sliko okrog dane zaslonske točke in za vsako prebrano globino, ki je nižja od globine dane zaslonske točke (torej bližje kameri), poveča števec. Po končanem vzorčenju za izračunano vrednost $A_{\mathbf{p}}$ vzamemo povprečno vrednost vseh vzorcev. Za preprečevanje popačenj zaradi bižnjih in preveč oddaljenih površin smo dodali še minimalno in maksimalno vrednost a in R_Ω. Delovanje je podobno vrednotenju integrala $A_{\mathbf{p}}$, le da se izvaja prek globinske slike. Rezultat efekta je prikazan na sliki 2.17.

[5] `http://john-chapman-graphics.blogspot.com/2013/01/ssao-tutorial.html`



Slika 2.18: Primerjava izrisov brez uporabe (levo) in z uporabo (desno) efekta globinske ostrine.

2.8.2 Globinska ostrina

En izmed načinov simuliranja leče kamere ali očesa je tudi efekt globinske ostrine. Leča lahko natančno izostri le predmete na fokusni razdalji, zato ostali elementi na končni sliki izgledajo zamegljeni oz. izven fokusa. S tako simulacijo naravnega pojava lahko ustvarimo bolj verodostojno sliko, hkrati pa izboljšamo uporabnikovo dojemanje globine. Nekoliko poenostavljen vizualni efekt globinske ostrine lahko izračunamo tudi v slikovnem prostoru. V ta namen algoritem uporablja tri slike: ostro končno sliko, zamegljeno končno sliko in globinsko sliko. Glede na oddaljenost globine iz globinske slike od fokusne razdalje se interpolira med zamegljeno in ostro sliko. Občutek globine lahko še dodatno poudarimo z zatemnitvijo slike glede na oddaljenost od fokusne razdalje. Tako dobimo sliko, ki je ostra in svetla na fokusni razdalji, drugod pa je temnejša in nekoliko zamegljena. Rezultat efekta globinske ostrine je prikazan na sliki 2.18. Opisani pristopi pozitivno vplivajo na dojemanje globine in hkrati učinkovito preusmerijo pozornost uporabnika na elemente v fokusu, kar pa v določenih primerih ni zaželeno, saj lahko globinska ostrina zabriše relevantne informacije.

Poglavje 3

Evalvacija in rezultati

V okviru predstavljenega dela smo razvili aplikacijo, ki vsebuje implementacije predstavljenih metod. Aplikacija je sestavljena iz različnih delov in je večinoma napisana v programskem jeziku Java. Glavni del aplikacije predstavlja program za vizualizacijo volumetričnih podatkov. Program za izris računalniške grafike uporablja programsko knjižnico LWJGL (Lightweight Java Game Library), ki omogoča komunikacijo z grafično strojno opremo preko vmesnikov OpenGL in OpenCL. Vmesnik OpenCL izkoriščamo za izrabo grafične procesne enote (GPE) za hiter izračun implementacije predstavljenih metod, vmesnik OpenGL pa uporabljamo zgolj za končni izris na zaslon.

Za ocenjevanje zmogljivosti in praktične uporabnosti naše aplikacije smo izvedli tudi performančno in uporabniško evalvacijo. Okolje in postopki, s katerimi sta bili evalvaciji izvedeni, so opisani v naslednjih poglavjih. V poglavjih 3.2 in 3.4 predstavimo rezultate evalvacij in njihovo analizo.

3.1 Performančna evalvacija

Vse performančne teste smo izvajali na namiznem računalniku z naslednjimi specifikacijami:

- Procesor: 2 procesorja Intel Xeon E5-2620 2.0 GHz,

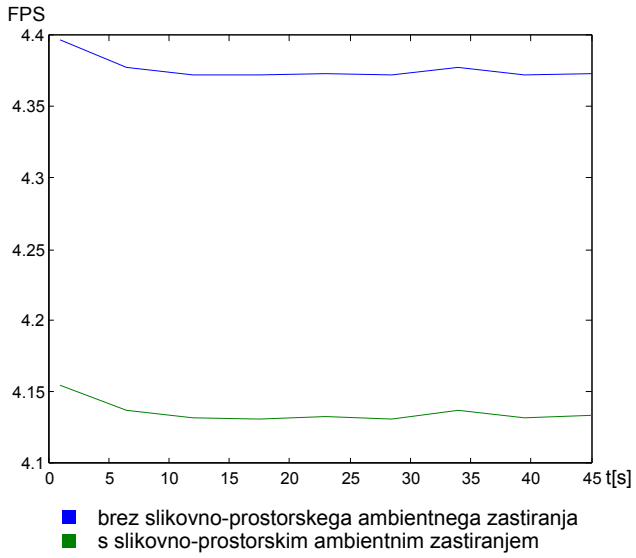
- Pomnilnik: 32 GB,
- Operacijski sistem: Windows Server 2008 R2 Standard,
- Grafična kartica: NVIDIA Quadro 2000 1 GB RAM.

Pri različnih metodah vizualizacije in položajih kamere smo spremljali hitrost izrisa slike. Statični testi so se izvajali v ločljivosti 1024×1024, animirani pa v ločljivosti 512×512. Upodabljali smo volumetrične podatke `mrt16_angio`, ki so dostopni na spletni strani `http://www.volvis.org/`. V grafih je predstavljena hitrost upodabljanja, ki smo jo merili v številu slik na sekundo. Meritve števila slik smo opravljali vsakih 5 sekund, da smo dobili povprečne vrednosti, ki smo jih nato uporabili v grafih. Statične teste smo izvajali z enakim položajem kamere, kot je vidno na sliki 2.9. Pri animiranih testih je kamera sledila krožni tirnici okrog centralne točke volumna na najmanjši možni razdalji, kjer še ni bila znotraj očrtanega kvadra (angl. bounding box) volumna. Hitrost kroženja je bila 0.1 radian na sekundo, teste pa smo izvajali 100 sekund. Ker je bil volumen različnih dimenzij, je na grafih animiranih testov jasno vidno nihanje hitrosti izrisa. Nižje hitrosti so bile zabeležene takrat, ko je bila kamera usmerjena vzdolž daljše stranice volumna in so žarki morali potovati dlje.

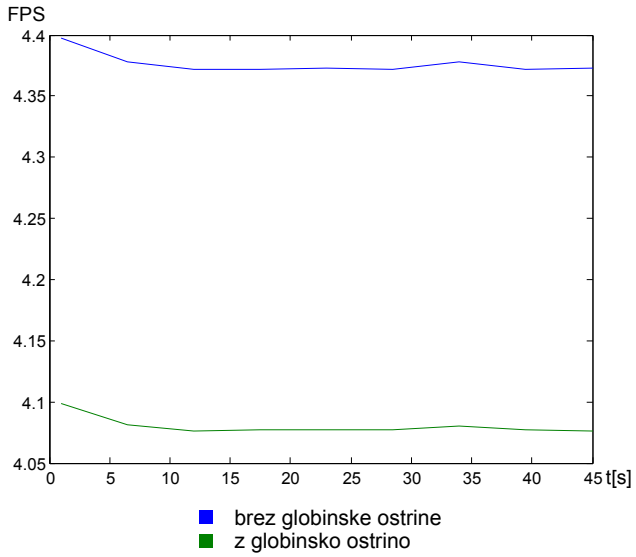
3.2 Rezultati performančne evalvacije

Grafa 3.1 in 3.2 prikazujeta hitrosti efektov slikovno-prostorkega ambientnega zastiranja in globinske ostrine v primerjavi s hitrostjo aplikacije, ko teh efektov nismo vključili. Efekta sta slikovno-prostorska, zato porabita izredno malo računskega časa. Grafi nakazujejo, da je aplikacija z uporabo efektov hitrejša, kar pa je bolj verjetno posledica večje obremenjenosti grafične kartice v času izvajanja testov brez efektov.

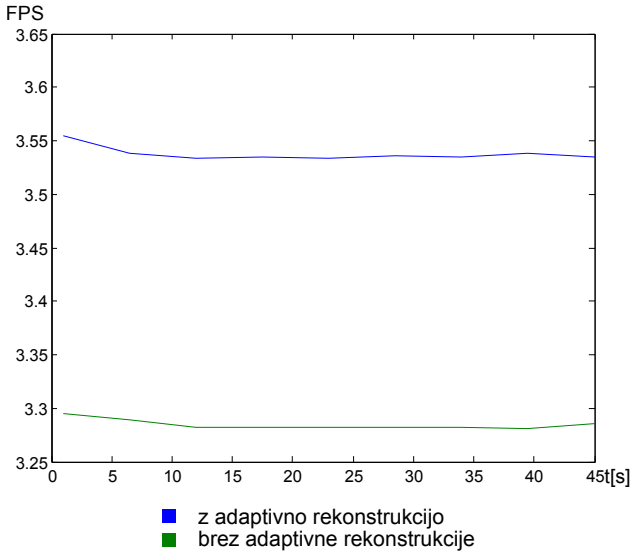
Na grafu 3.3 je jasno vidno, da naša metoda adaptivne rekonstrukcije pozitivno vpliva na hitrost upodabljanja. Pohitritev znaša okrog 8%. Za kar-



Slika 3.1: Hitrost upodabljanja z uporabo slikovno-prostorkega ambientnega zastiranja in brez njegove uporabe.



Slika 3.2: Hitrost upodabljanja z uporabo efekta globinske ostrine in brez njegove uporabe.

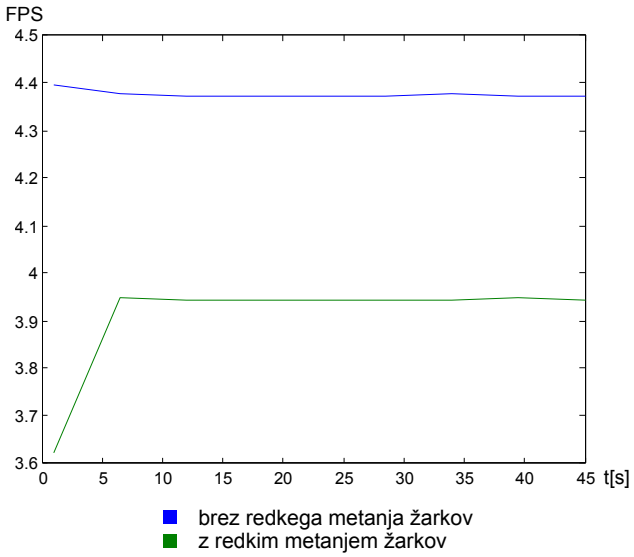


Slika 3.3: Hitrost upodabljanja z uporabo adaptivne rekonstrukcije in brez njegove uporabe.

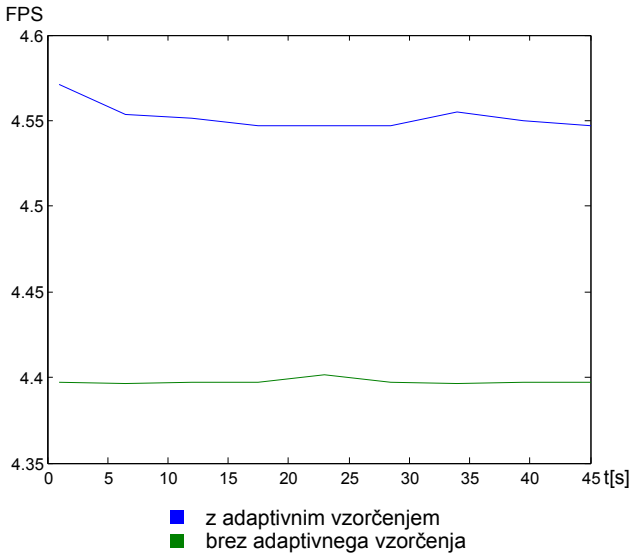
seda nepristransko primerjavo smo pri tem testu izključili uporabo osmiškega drevesa.

V nasprotju s pričakovanji je hitrost izrisa počasnejša z uporabo metode redkega metanja žarkov (graf 3.4). To je verjetno posledica neoptimizirane OpenCL kode, ki bi jo lahko še precej izboljšali. Prav tako smo v naši aplikaciji implementirali le en nivo redkega metanja žarkov, kar bi lahko v prihodnosti razširili s polno implementacijo metode. Čeprav je število žarkov mnogo manjše, pa je zato toliko več vzorčenja vrednosti barv na vmesni sliki v drugi fazi algoritma. Tam je prisotnih tudi več pogojnih stavkov, ki pa so neprimerni za izvajanje na grafični kartici.

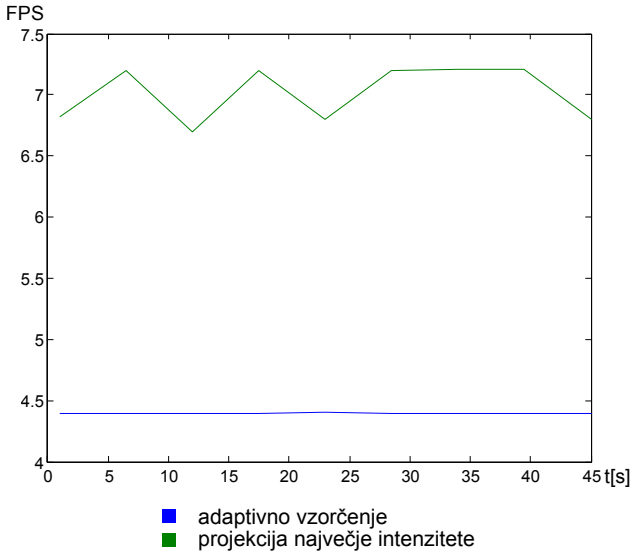
Naša metoda adaptivnega vzorčenja, ki smo jo razvili za hitrejše delovanje upodabljanja prosojnih materialov, se je izkazala za uspešno. Primerjava, vidna na grafu 3.5, prikazuje pohitritev izrisa za približno 4% v primerjavi z enakomernim vzorčenjem. Ker so pohitritve zelo odvisne od globine volumna in funkcije adaptivnega vzorčenja, bi lahko s primernejšo izbiro parametrov funkcije dobili tudi boljše rezultate.



Slika 3.4: Hitrost upodabljanja z uporabo redkega metanja žarkov in brez njegove uporabe.



Slika 3.5: Hitrost upodabljanja z uporabo adaptivnega vzorčenja in brez njegove uporabe.



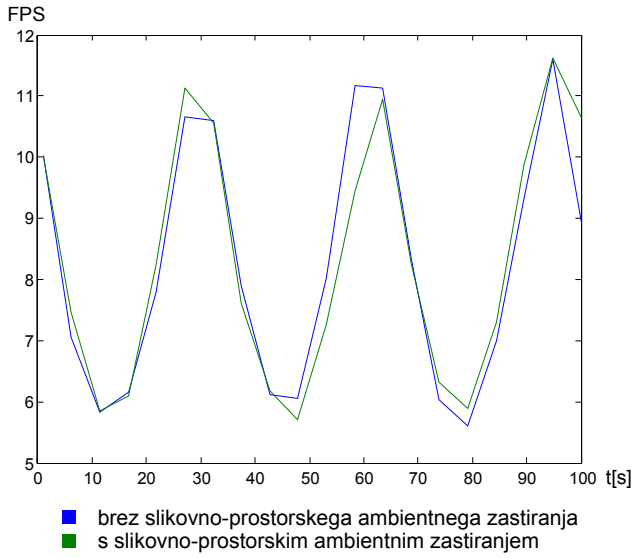
Slika 3.6: Hitrost upodabljanja z uporabo projekcije največje intenzitete v primerjavi z adaptivnim vzorčenjem.

Glede hitrosti izrisa najbolj izstopa metoda projekcije največje intenzitete (vidno na grafu 3.6). Kot smo že omenili v poglavju 2.5.5, zaradi hitrosti izrisa trpi kvaliteta končne slike.

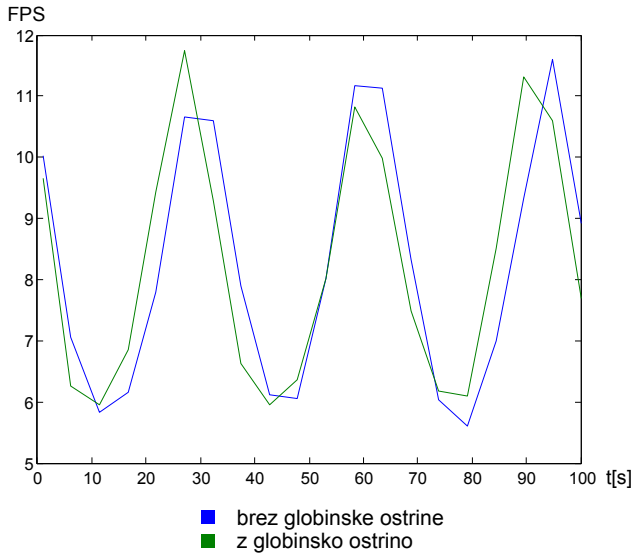
Sliki 3.7 in 3.8 prikazujeta uporabo vizualnih efektov slikovno-prostorskega ambientnega zastiranja in globinske ostrine v animiranem testu. Grafa kažeta le manjša odstopanja, zato sklepamo, da efekta porabita zanemarljivo malo računskega časa.

Adaptivna rekonstrukcija (graf 3.9) se je v animiranem testu dobro odrezala, predvsem tam, kjer je bila kamera usmerjena vzdolž daljše stranice volumna. Takrat je bilo namreč potrebnih več vzorčenj volumna tudi v praznem prostoru.

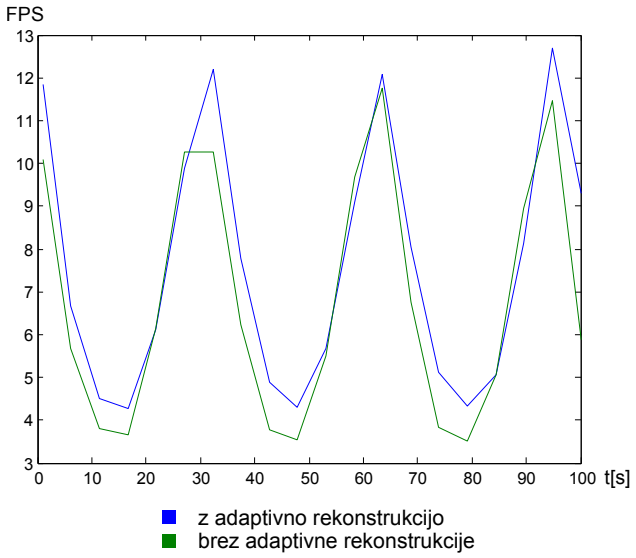
V animiranem testu se je bolje odrezala metoda redkega metanja žarkov. Na grafu 3.10 lahko vidimo, da so bile pohitritve večje v primerih, ko je bila kamera usmerjena vzdolž krajše stranice volumna. V tem primeru je bila prva faza algoritma krajša, zato je imela druga faza sorazmerno večji učinek na hitrost. Pohitritev je vidna tudi drugod, toda v manjši meri.



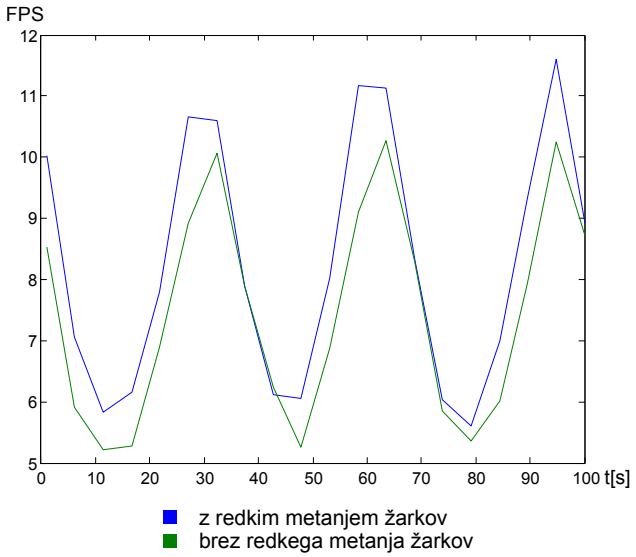
Slika 3.7: Hitrost upodabljanja z uporabo slikovno-prostorkega ambientnega zastiranja in brez njegove uporabe.



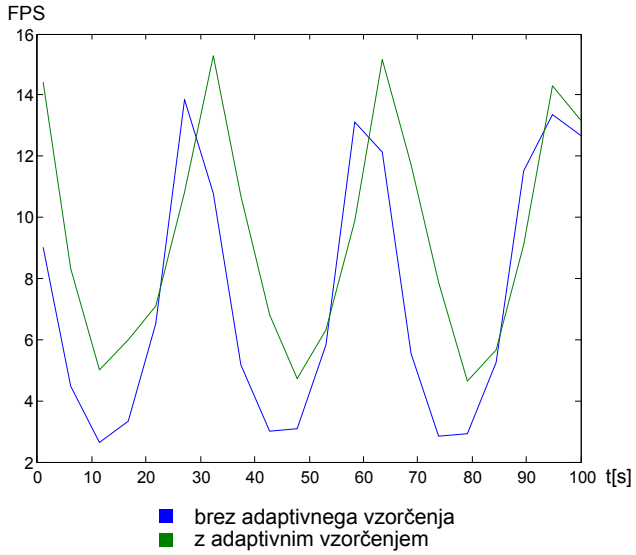
Slika 3.8: Hitrost upodabljanja z uporabo efekta globinske ostrine in brez njegove uporabe.



Slika 3.9: Hitrost upodabljanja z uporabo adaptivne rekonstrukcije in brez njegove uporabe.



Slika 3.10: Hitrost upodabljanja z uporabo redkega metanja žarkov in brez njegove uporabe.

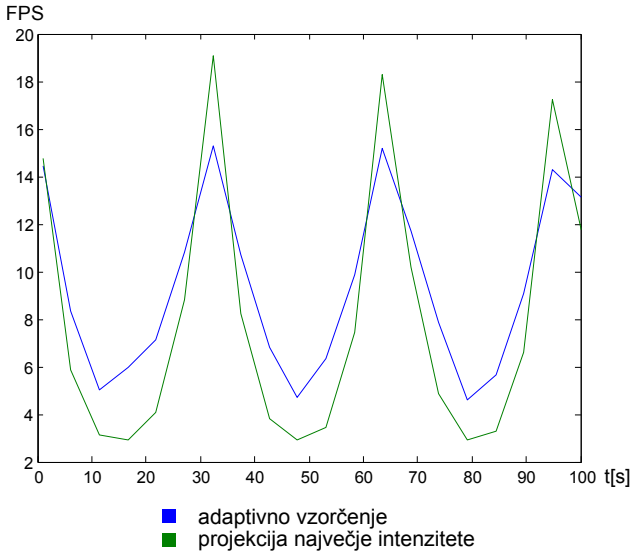


Slika 3.11: Hitrost upodabljanja z uporabo adaptivnega vzorčenja in brez njegove uporabe.

Graf 3.11, ki prikazuje hitrost adaptivnega vzorčenja, razkriva velike pohitritve. Metoda sicer porabi nekoliko več časa za računanje dolžine koraka, toda pri daljših korakih so dobički na hitrosti v primerjavi z izgubami precej večji.

Graf hitrosti projekcije največje intenzitete v animiranem testu (graf 3.12) nam ponovno prikazuje velike hitrosti te metode. V primerjavi z adaptivnim vzorčenjem je precej počasnejša v primeru, ko je globina volumna iz pogleda kamere večja. Takrat namreč metoda adaptivnega vzorčenja primerno zmanjša količino vzorčenj, metoda projekcije največje intenzitete pa nadaljuje z enakomernim vzorčenjem.

Hitrost vseh metod z izjemo projekcije največje intenzitete je medsebojno primerljiva, toda v primerjavi z nekaterimi obstoječimi implementacijami še vedno nizka. Razlog zelo verjetno tiči v temeljni uporabljeni tehnologiji. Večina obstoječih aplikacij namreč uporablja jezike, ki so namenjeni programiranju na grafičnih karticah, kot sta GLSL (angl. OpenGL shading language) in HLSL (angl. high-level shading language), medtem ko smo v naši



Slika 3.12: Hitrost upodabljanja z uporabo projekcije največje intenzitete v primerjavi z adaptivnim vzorčenjem.

implementaciji uporabili jezik OpenCL. Prednost tega pristopa je v prenosljivosti kode, saj je v tem primeru mogoče računati na zelo različnih napravah, medtem ko sta jezika GLSL in HLSL namenjena izvajanju na grafičnih karticah.

3.3 Uporabniška evalvacija

V okviru tega dela smo izvedli tudi uporabniško evalvacijo različnih načinov vizualizacije volumetričnih podatkov z namenom iskanja zaželenih lastnosti vizualizacijskih metod. Glavni cilj evalvacije je pridobiti mnenje o ustreznosti uporabe posameznih vizualizacijskih metod z njihovo medsebojno primerjavo z namenom izboljšave hitrosti, kvalitete in zanesljivosti procesov diagnosticiranja ter odkrivanja anevrizem in drugih nepravilnosti na krvožilnem sistemu. Namenjena je bila predvsem specialistom radiologije, saj so potencialni uporabniki naše aplikacije. Evalvacijo smo zasnovali v obliki spletnega vprašalnika s primerjalnimi testi. Od uporabnikov smo pričakovali

Slika 3.13: Spletna aplikacija za izvedbo uporabniške evalvacije.

vnos osnovnih demografskih informacij: starosti, izobrazbe, od zdravnikov tudi specializacije, nato pa smo jih pozvali k ocenjevanju parov končnih slik. V fazi ocenjevanja slik so lahko slike povečali in si jih bolje ogledali, nato pa za določen par slik izbrali tisto sliko, za katero so menili, da jim omogoča lažjo in hitrejšo prepoznavo oblik, površin, globine, medsebojnih razdalj in prekrivanja predmetov. Preferenc seveda nismo zahtevali, saj so lahko uporabniki za določen par slik ostali neodločeni. Spletni vprašalnik je prikazan na sliki 3.13.

Slike so prikazovale tri različne medicinske volumne, vsak volumen pa je bil prikazan z dveh strani. Med seboj smo primerjali 6 tehnik vizualizacije:

- prikaz nivojskih ploskev z algoritmom marching cubes (MC),
- prikaz nivojskih ploskev z algoritmom MPUI s kockami (CUBE),
- prikaz nivojskih ploskev z algoritmom MPUI s tetraedri (TET),

- prikaz nivojskih ploskev z metanjem žarkov s pravim in slikovno-prostorskim ambientnim zastiranjem (ISO),
- prikaz nivojskih ploskev z metanjem žarkov s pravim in slikovno-prostorskim ambientnim zastiranjem ter z globinsko ostrino (DOF),
- upodabljanje prosojnih materialov s simulacijo emisijsko-absorpcijskega optičnega modela (ALPHA).

Iz množice teh 6 metod lahko izberemo $\binom{6}{2} = 15$ različnih parov - za dva pogleda na treh različnih volumnih to pomeni 90 parov slik. Mešanja različnih pogledov ali različnih volumnov v primerjalnih testih namreč nismo dopustili. Algoritma marching cubes in MPUI je implementiral Miloš Lukič v okviru svoje diplomske naloge [22]. V oklepajih so zapisane oznake tehnik, ki so uporabljene tudi v tabeli 3.1.

3.4 Rezultati uporabniške evalvacije

Za analizo rezultatov smo uporabili metodo BRE (angl. balanced rank estimation), opisano v [45], in metodo glasovanja VOTE. Pri prvi metodi je izbrana vizualizacijska tehnika dobila glas, neizbrana pa je glas izgubila. Pri metodi glasovanja je le izbrana vizualizacijska tehnika dobila glas. V obeh primerih je število zbranih glasov določalo končni vrstni red, pri katerem je višje število pomenilo boljšo uvrstitev.

Postopek analize smo izvedli na 6 anketah, ki so bile izpolnjene v celoti. Te ankete je rešilo 6 radiologov, starih od 30 do 48 let, s povprečno starostjo 38.67 let. Poleg analize popolnih anket smo izvedli tudi analizo vseh anket, kjer je prišlo do minimalnih odstopanj, toda vrstni red je večinoma ostal nespremenjen. Razlike so se pojavile le pri analizi VOTE, kjer se je metoda izrisa nivojskih ploskev z efektom globinske ostrine (DOF) uvrstila pred metodi MPUI (CUBE in TET). V tabeli 3.1 so zbrane metode z ocenami analize, razvrščeni od najboljše do najslabše po metodi BRE. Oznaki

	BRE	VOTE	BRE*	VOTE*
ISO	67	103	86	128
MC	11	71	15	87
CUBE	10	66	7	79
TET	-7	57	-12	66
DOF	-22	61	-15	80
ALPHA	-59	41	-81	65

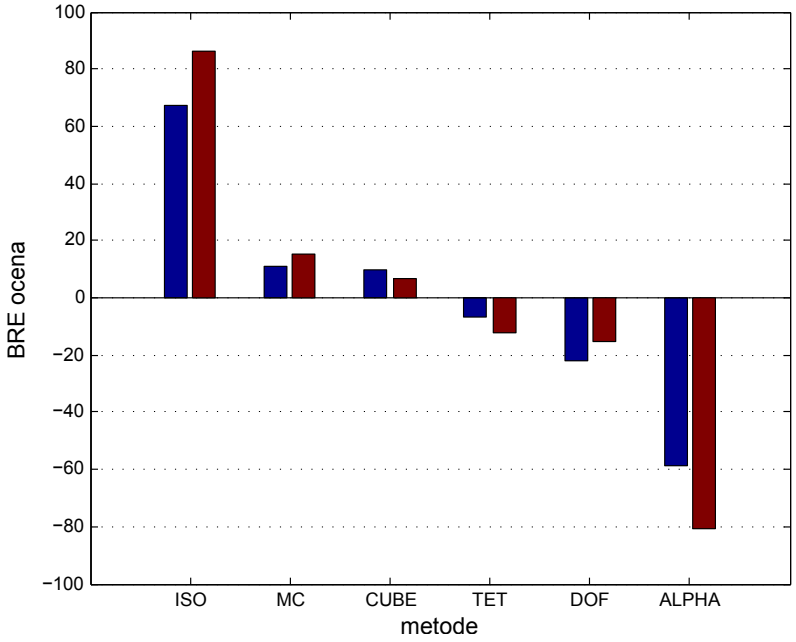
Tabela 3.1: Rezultati BRE in VOTE analize uporabniške evalvacije. Oznake metod so razložene v poglavju 3.3. Oznaki BRE* in VOTE* označujeta analizo vseh anket, ne le popolnih.

BRE* in VOTE* označujeta analizo vseh anket, ne le popolnih. Rezultati so prikazani vizualno na slikah 3.14 in 3.15.

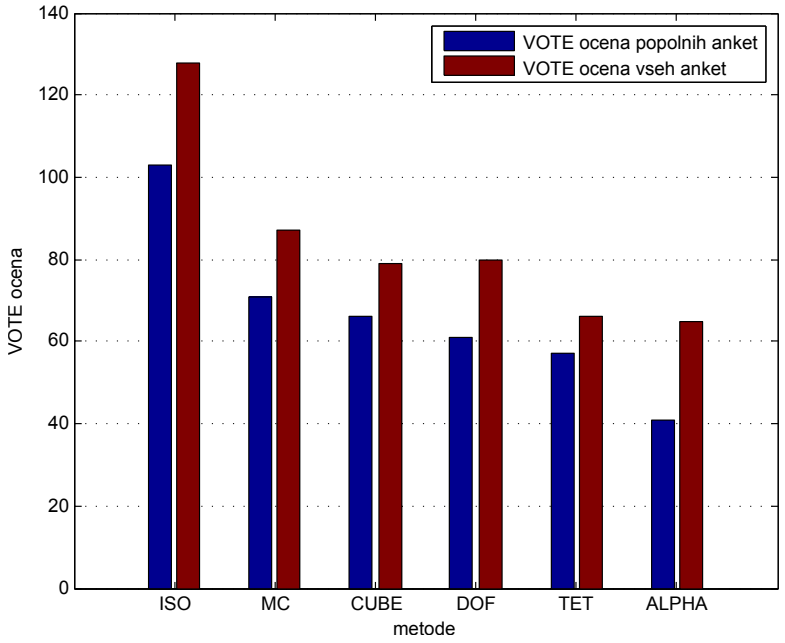
Nedvomni zmagovalec je metoda metanja žarkov s pravim in slikovno-prostorskim ambientnim zastiranjem (ISO), medtem ko so se ostale metode odrezale primerljivo dobro. Med slabšimi izstopa še simulacija emisijsko-absorpcijskega optičnega modela (ALPHA). Na njeno slabo uvrstitev je zelo verjetno vplivala slaba izbira klasifikacijskega gradienta, ki je bil razlog za nekoliko nekonstrastno sliko. Ker je bila slika z efektom globinske ostrine (DOF) tudi temnejša, je zelo verjetno to negativno vplivalo na njeno uvrstitev.

Na slikah 3.16 in 3.17 je prikazana analiza parov slik za popolne ankete oz. za vse ankete. S slike lahko razberemo kako so se posamezne metode odrezale v primerjavi z drugimi metodami. Matriki sta antisimetrični.

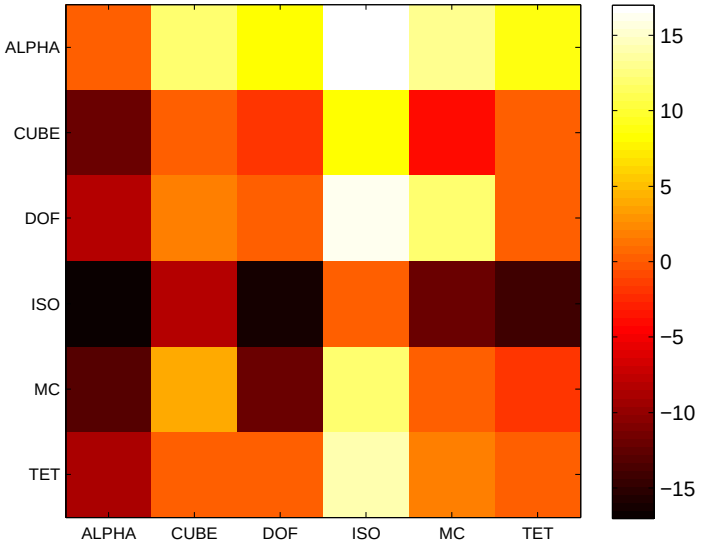
S slik je razvidna velika premoč metode ISO v primerjavi z ostalimi metodami. Metode MC, CUBE in TET so tudi glede na rezultate analize parov primerljivo dobre. Medsebojna primerjava teh treh metod pa kaže, da se CUBE uvršča pred MC, medtem ko sta MC in TET povsem primerljivi. Rezultati, pridobljeni z analizo vseh anket, so sicer z manjšimi odstopanji podobni rezultatom analize popolnih anket.



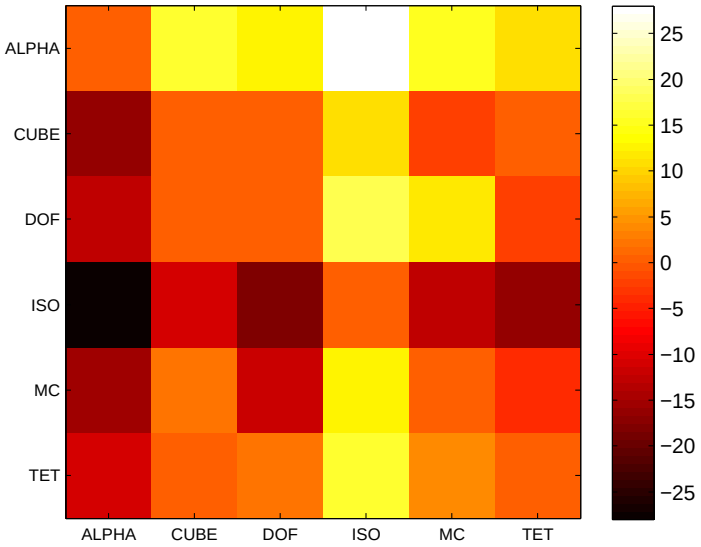
Slika 3.14: Analiza uporabniške evalvacije z metodo BRE.



Slika 3.15: Analiza uporabniške evalvacije z metodo VOTE.



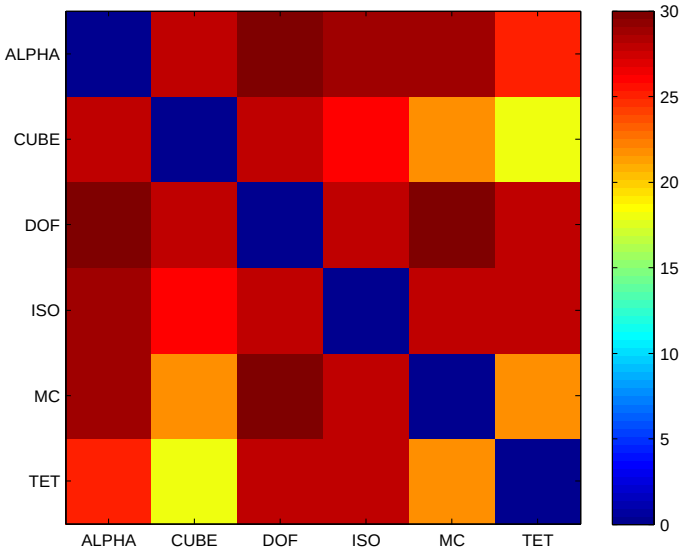
Slika 3.16: Analiza parov za popolne ankete.



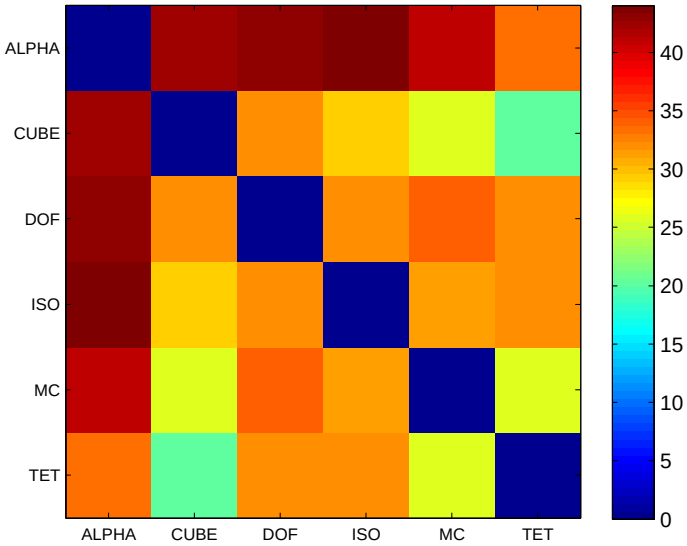
Slika 3.17: Analiza parov za vse ankete.

Na slikah 3.18 in 3.19 je prikazana analiza neodločenosti za popolne ankete oz. za vse ankete. Nižja vrednost pomeni, da so uporabniki večkrat ostali neodločeni pri dani izbiri, kar zelo nazorno prikazujejo diagonale, kjer so prikazane primerjave metod samih s seboj, česar v anketi nismo dopustili. Matriki sta simetrični.

Kot je razvidno iz omenjenih slik so bili uporabniki najbolj neodločeni pri primerjavi metod TET in CUBE, kar smo pričakovali. V obeh primerih gre namreč za metodo MPUI, le da sta uporabljena različna algoritma za poligonizacijo. Visoka stopnja neodločenosti je prisotna tudi pri primerjavi metode MC z metodama TET in CUBE. Iz rezultatov lahko vidimo, da uporabniki niso imeli težav pri izbiri med metodama DOF in ALPHA oz. DOF in MC. Preseneča predvsem preferenca uporabnikov za izbiro metode DOF pred metodo ALPHA. Ta izid pripisujemo nerazumevanju slik, pridobljenih z metodo ALPHA, in neprimerni izbiri klasifikacijskega gradienta. Prav tako uporabniki niso imeli težav pri odločanju, kjer sta bili v izbiri prisotni metodi ISO ali ALPHA. Na sliki 3.19 izstopa metoda ALPHA, kar kaže na pogostejšo pojavitev te metode v parih slik.



Slika 3.18: Analiza neodločenosti za popolne ankete.



Slika 3.19: Analiza neodločenosti za vse ankete.

Poglavje 4

Zaključki in nadaljnje delo

V tem delu smo predstavili številne metode in tehnike za vizualizacijo medicinskih volumetričnih podatkov. Glavni trije načini vizualizacije so upodabljanje nivojskih ploskev, projekcija največje intenzitete in upodabljanje prosojnih materialov. Vsak od teh načinov ima seveda svoje prednosti in slabosti, ki jih bomo predstavili v nadaljnjem besedilu. Podrobnejše obravnave je deležna tudi podporna tehnologija, ki skrbi za hranjenje podatkov in izvajanje poizvedb.

Predstavitev in shranjevanje podatkov se tiče vseh načinov vizualizacije. V naši implementaciji smo celoten volumen shranili v pomnilnik grafične kartice, nad njim pa zgradili polno osmiško drevo. Glavna prednost tega pristopa je hitrost poizvedb in vzorčenja, posledično pa bistveno hitrejši celoten postopek vizualizacije. Ker je natančnost izrisa omejena z natančnostjo volumna, je največja omejitev naše implementacije odvisnost od velikosti pomnilnika grafične kartice. V ta namen nameravamo v prihodnosti našo aplikacijo izboljšati s sistemom za upravljanje s pomnilnikom, ki bo podatke po potrebi bral z diska. Tako bi rešili težavo z velikostjo pomnilnika, boriti pa bi se morali s hitrostjo izrisa, saj bi podporni postopki za ažuriranje podatkov zahtevali precej več računskega časa. Količino podatkov bi lahko zmanjšali tudi z implementacijo redkega osmiškega drevesa, prav tako na račun trajanja vizualizacije.

Prvi način vizualizacije je upodabljanje nivojskih ploskev. Za izboljšavo kvalitete končne slike smo implementirali metodo regula falsi za natančnejše določanje presečišč žarkov z nivojsko ploskvijo, kar je tudi bistveno pripomoglo h kvaliteti senčenja. Boljše rezultate bi v prihodnosti lahko dosegli z Newtonovo metodo, toda potrebovali bi hitrejšo implementacijo. V postopku senčenja naša implementacija uporablja Phongov model, katerega bi v prihodnosti lahko zamenjali z bolj realnim modelom, denimo s Cook-Torranceovim. Višjo hitrost izrisa smo dosegli s preskakovanjem praznega prostora, kjer nam je koristilo osmiško drevo. Čas vzorčenja v praznem prostoru smo dodatno skrajšali z novo pohitritveno metodo: adaptivno rekonstrukcijo. Z uporabo te metode smo uspeli zmanjšati kompleksnost vzorčenja v praznem prostoru in ohraniti enako kvaliteto končne slike. Hitrost izrisa smo tako povečali za 8%. Število potrebnih žarkov smo na najbolj neposreden način uspeli zmanjšati z uporabo metode redkega metanja žarkov. Zaradi vzpodbudnih rezultatov nameravamo v prihodnosti implementirati celotno različico metode.

Po uspešni implementaciji upodabljanja nivojskih ploskev smo se odločili, da preizkusimo tudi metode za upodabljanje prosojnih materialov. Najenostavnejša metoda, ki pa je v praksi dokaj pogosta, je projekcija največje intenzitete (MIP). Metodo smo uspešno priredili za delovanje v realnem času tudi za zaslone večjih ločljivosti. Hitrost smo povečali tudi z uporabo adaptivne rekonstrukcije. Na račun hitrosti izrisa trpi kvaliteta končne slike, saj izgubimo veliko informacij o volumnu, ki so sicer razpoložljive pri drugih načinih vizualizacije. Za izboljšavo kvalitete slike smo dodali še nizek prag, ki nam je omogočal enostavno izločanje šuma. Dodatna izboljšava, ki jo nameravamo implementirati v prihodnosti, je projekcija lokalne največje intenzitete (LMIP). Skupaj z animacijo bi na ta način bistveno izboljšali dojemanje globine in medsebojnih razdalj v prostoru.

Največji napredek v smislu kvalitete vizualizacije smo dosegli s simulacijo transporta svetlobe z emisijsko-absorpcijskim modelom. Metoda izrisuje materiale, katerim lahko uporabnik sam določa različne optične lastnosti, na

primer barvo in prosojnost. Naša aplikacija te optične lastnosti, zajete v preslikovalni tabeli, prebere z diska ob zagonu. To bi lahko enostavno razširili s podporo dinamičnemu spreminjanju preslikovalne tabele tekom izvajanja aplikacije oz. z integracijo interaktivnega pripomočka za gradnjo gradientov v samo aplikacijo. Prav zaradi teh preslikovalnih tabel je metoda zmožna upodabljanja tako polne notranjosti žil kot le žilnih sten. Žal pa je zaradi svoje napredne zasnove tudi nekoliko počasnejša, kar se bistveno pozna pri večjih volumnih. To težavo smo do neke mere uspeli premostiti z novo metodo adaptivnega vzorčenjam, ki hitrost izrisa poveča za 4%, in osmiškim drevesom za preskakovanje praznih oz. homogenih območij volumna. Kljub temu je dojemanje globine in obnašanja svetlobe na površinah materialov precej slabše kot pri upodabljanju nivojskih ploskev, kar je prišlo do izraza v uporabniški evalvaciji. To bi lahko v prihodnosti rešili z vpeljavo naprednih tehnik senčenja in osvetljevanja v optični model, kar bi hkrati povzročilo tudi počasnejšo vizualizacijo. Za natančnejše reševanje integrala volumetričnega upodabljanja bomo v prihodnosti preizkusili delovanje Rombergove metode in metod Runge-Kutta.

Končno sliko smo uspeli izboljšati tudi z vizualnimi efekti. Implementirali smo slikovno-prostorsko ambientno zastiranje, ki skupaj s pravim ambientnim zastiranjem bistveno izboljša dojemanje topologije in medsebojnih razdalj v prostoru pri metodi upodabljanja nivojskih ploskev, prilagodili pa smo jo tudi za delovanje na prosojnih materialih, kjer dosega presenetljivo dobre rezultate. Poleg ambientnega zastiranja smo preizkusili tudi delovanje efekta globinske ostrine, ki ob zmerni uporabi pozitivno vpliva na dojemanje lastnosti volumna. Oba vizualna efekta delujeta v slikovnem prostoru, kar pripomore k visoki hitrosti izračunavanja. Ugotovili smo, da bi v mnogih primerih končno sliko lahko dodatno izboljšali še s tehnikami s področja obdelave fotografij kot so na primer visok dinamični razpon, preslikava barv ter popravki kontrastov, svetlosti in vrednosti gama.

Naša pričakovanja o primernosti vizualizacijskih metod je do neke mere potrdila tudi uporabniška evalvacija, ki smo jo izvedli v obliki spletnega

vprašalnika s primerjalnimi testi. Z metodama BRE in VOTE smo nato razvrstili metode po primernosti uporabe za diagnosticiranje angiografskih volumetričnih slik. Rezultati kažejo na veliko prednost upodabljanja nivojskih ploskev v primerjavi z emisijsko-absorpcijskim modelom. Po rezultatih analize sodeč sklepamo, da efekta pravega in slikovno-prostorskega ambientnega zastiranja pozitivno vplivata na kvaliteto končne slike. Efekt globinske ostrine se v analizi ni tako dobro odrezal, toda to pripisujemo neprimerni izbiri parametrov in moči efekta. Ostale metode so se odrezale primerljivo dobro, na kar nakazuje tudi analiza parov. Analiza neodločenosti uporabnikov pri izbiri slik kaže predvsem na enakovrednost metod marching cubes in obeh različic metode MPUI, prav tako pa je razvidno, da je metoda upodabljanja nivojskih ploskev z ambientnim zastiranjem nesporni zmagovalec.

Po obširni performančni analizi uporabljenih metod sklepamo, da so uporabljene metode primerne za interaktivno vizualizacijo, kljub temu pa je potrebno še veliko izboljšav, da bo interakcija potekala bolj gladko. Ker smo za implementacijo uporabili programski jezik Java in vmesnik OpenCL, je to zelo verjetno pripomoglo k počasnejšemu izvajanju algoritmov. Kjer bi s primernejšo izbiro tehnologij lahko pridobili na hitrosti, smo se raje odločili za prenosljivost programske kode. Tako Java kot OpenCL sta namreč zasnovana za izvajanje na čim širši paleti naprav.

Za potrebe hitrejše, kvalitetnejše in zanesljivejše diagnoze anevrizem imamo namen implementirati tudi metode prostorko spreminjajočega se senčenja. Želeli bi namreč poudariti tista območja volumna, kjer je verjetnost pojavitve anevrizme večja. Največja težava te ideje je v porabi pomnilnika grafične kartice, kar pa bi rešili s sistemom za upravljanje s pomnilnikom.

Največja omejitev trenutnih implementacij metod je hitrost upodabljanja, saj kvalitetno upodabljanje v visoki ločljivosti v realnem času še ni možno. Doseganje interaktivnih hitrosti je sicer možno na račun kvalitete izrisa ali ločljivosti končne slike, toda te omejitve tudi izredno negativno vplivajo na postopek diagnosticiranja. V prihodnosti se bomo posvetili optimizaciji uporabljenih algoritmov ter preizkušanju novejših in sodobnejših algoritmov.

Literatura

[1] James F. Blinn. Models of light reflection for computer synthesized pictures. *ACM SIGGRAPH Computer Graphics*, 11(2):192–198, Avgust 1977.

[2] Jules Bloomenthal. *An implicit surface polygonizer*, del 1. Academic Press Professional, Inc., 1994.

[3] Robert L. Cook in Kenneth E. Torrance. A reflectance model for computer graphics. *ACM SIGGRAPH Computer Graphics*, 15(3):307–316, Avgust 1981.

[4] Robert A. Drebin, Loren Carpenter, in Pat Hanrahan. Volume rendering. *ACM SIGGRAPH Computer Graphics*, 22(4):65–74, 1988.

[5] Tim Foley in Jeremy Sugerman. KD-tree acceleration structures for a GPU raytracer. Objavljeno v *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, del 45, strani 15–22, 2005.

[6] Andrew S. Glassner. Space subdivision for fast ray tracing. *IEEE Computer Graphics and Applications*, 4(10):15–24, 1984.

[7] Enrico Gobbetti, Fabio Marton, in José Antonio Iglesias Guitián. A single-pass GPU ray casting framework for interactive out-of-core rendering of massive volumetric datasets. *The Visual Computer*, 24(7-9):797–806, 2008.

[8] Sören Grimm, Stefan Bruckner, Armin Kanitsar, in Eduard Gröller. Memory efficient acceleration structures and techniques for CPU-based volume raycasting of large data. Objavljeno v *Proceedings IEEE/SIGGRAPH Symposium on Volume Visualization and Graphics*, del Vi, strani 1–8, 2004.

[9] John C. Hart. Ray tracing implicit surfaces. Objavljeno v *SIGGRAPH '93 Course Notes: Design, Visualization and Animation of Implicit Surfaces*, del 1, 1993.

[10] John C. Hart. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1996.

[11] Andrea Kratz. *Advanced illumination techniques for GPU-based direct volume rendering.* Diplomsko delo, University of Koblenz-Landau, 2006.

[12] Andrea Kratz, Jan Reininghaus, Markus Hadwiger, in Ingrid Hotz. Adaptive screen-space sampling for volume ray-casting. Technical Report February, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2011.

[13] Philippe Lacroute in Marc Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. Objavljeno v *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, strani 451–458, 1994.

[14] Samuli Laine in Tero Karras. Efficient sparse voxel octrees. *IEEE Transactions on Visualization and Computer Graphics*, 17(8):1048–1059, 2011.

[15] Michael S. Langer in Heinrich H. Bülthoff. Depth discrimination from shading under diffuse lighting. *Perception*, 29(6):649–660, 2000.

[16] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):22–37, 1988.

[17] Marc Levoy. A hybrid ray tracer for rendering polygon and volume data. *IEEE Computer Graphics and Applications*, 10(2):33–40, 1990.

[18] Marc Levoy. Volume rendering by adaptive refinement. *The Visual Computer*, 6(1):2–7, 1990.

[19] Feng Ling in Ling Yang. Improved on maximum intensity projection. Objavljeno v *Proceedings of the 2009 International Conference on Artificial Intelligence and Computational Intelligence*, del 4, strani 491–495, 2009.

[20] Tom Lokovic in Eric Veach. Deep shadow maps. Objavljeno v *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, strani 385–392, New York, New York, USA, 2000. ACM Press.

[21] William E. Lorensen in Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, 1987.

[22] Miloš Lukić. *Primerjava algoritmov za rekonstrukcijo 3D modelov iz volumetričnih podatkov*. Diplomsko delo, Univerza v Ljubljani, 2014.

[23] Daniel Madeira, Esteban Clua, in Thomas Lewiner. GPU octrees and optimized search. Objavljeno v *Proceedings of the 8th Brazilian Symposium on Games and Digital Entertainment*, strani 73–76, 2009.

[24] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, Junij 1995.

[25] Donald Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):85, 1982.

[26] Gavin Miller. Efficient algorithms for local and global accessibility shading. Objavljeno v *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, strani 319–326, 1994.

[27] Lukas Mroz, Andreas König, in Eduard Gröller. Maximum intensity projection at warp speed. *Computers & Graphics*, 24(3):343–352, Junij 2000.

[28] Fred E. Nicodemus. Directional Reflectance and Emissivity of an Opaque Surface. *Applied Optics*, 4(7):767–775, Julij 1965.

[29] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, in Hans-Peter Seidel. Multi-level partition of unity implicits. *ACM Transactions on Graphics*, 22(3):463–470, Julij 2003.

[30] Michael Oren in Shree K. Nayar. Generalization of Lambert's reflectance model. Objavljeno v *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, strani 239–246, New York, New York, USA, 1994. ACM Press.

[31] Mark Owen. *Practical signal processing*. Cambridge University Press, illustrate edition, 2012.

[32] Daniel P. Petersen in David Middleton. Sampling and reconstruction of wave-number-limited functions in N-dimensional euclidean spaces. *Information and Control*, 5(4):279–323, December 1962.

[33] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, Junij 1975.

[34] Thomas Porter in Tom Duff. Compositing digital images. *ACM SIGGRAPH Computer Graphics*, 18(3):253–259, Julij 1984.

[35] Jorge Revelles, Carlos Ureña, in Miguel Lastra. An efficient parametric algorithm for octree traversal. Objavljeno v *Journal of WSCG*, del 8, strani 212–219, 2000.

[36] Kristof Römisch. *Sparse voxel octree ray tracing on the GPU*. Magistrsko delo, Aarhus University, 2009.

[37] Scott D. Roth. Ray casting for modeling solids. *Computer Graphics and Image Processing*, 18(2):109–144, Februar 1982.

[38] Yoshinobu Sato, Nobuyuki Shiraga, Shin Nakajima, Shinichi Tamura, in Ron Kikinis. Local maximum intensity projection (LMIP): A new rendering method for vascular visualization. *Journal of Computer Assisted Tomography*, 22(6):912–917, 1998.

[39] Mathias Schott, Vincent Pegoraro, Charles Hansen, Kévin Boulanger, in Kadi Bouatouch. A directional occlusion shading model for interactive direct volume rendering. *Computer Graphics Forum*, 28(3):855–862, Junij 2009.

[40] Claude E. Shannon. Communication in the presence of noise. *Proceedings of the IEEE*, 86(2):447–457, Februar 1998.

[41] Anže Sodja. *Segmentacija prostorskih medicinskih podatkov na GPE*. Diplomsko delo, Univerza v Ljubljani, 2013.

[42] Marko Tatić. *Odloženo upodabljanje*. Diplomsko delo, Univerza v Ljubljani, 2013.

[43] Allen Van Gelder in Kwansik Kim. Direct volume rendering with shading via three-dimensional textures. Objavljeno v *Proceedings of the 1996 symposium on Volume visualization*, strani 23–30. ACM, 1996.

[44] Veronika Šoltészová, Daniel Patel, Stefan Bruckner, in Ivan Viola. A multidirectional occlusion shading model for direct volume rendering. *Computer Graphics Forum*, 29(3):883–891, Avgust 2010.

[45] Fabian L. Wauthier, Michael I. Jordan, in Nebojsa Jojic. Efficient ranking from pairwise comparisons. *Journal of Machine Learning Research*, 28(3):109–117, 2013.

[46] Felix Weißig. *Erstellung, Segmentierung und Out-Of-Core-Speichermanagement von Sparse Voxel Octrees.* Diplomsko delo, Bauhaus-Universität Weimar, 2013.

[47] Lee Alan Westover. *Splatting: a parallel, feed-forward volume rendering algorithm.* Doktorska disertacija, University of North Carolina, 1991.

[48] Ilmi Yoon, Joe Demers, Taeyong Kim, in Ulrich Neumann. Accelerating volume visualization by exploiting temporal coherence. Objavljeno v *Proceedings of IEEE Visualization*, strani 21–24, 1997.

[49] Yubo Zhang, Zhao Dong, in Kwan-Liu Ma. Real-time volume rendering in dynamic lighting environments using precomputed photon mapping. *IEEE Transactions on Visualization and Computer Graphics*, 19(8):1317–1330, Avgust 2013.